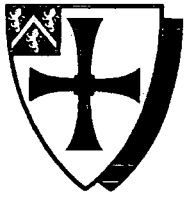
University of Durham



Belief Systems

for

Persuasive Discourse Planning

Massimiliano Garagnani

Ph.D. Thesis

Laboratory for Natural Language Engineering
Department of Computer Science

1999

To my Grandfather

Acknowledgments

When, back in 1994, I first saw, on the departmental board of the University of Bologna, the notice publicizing the research which was being carried out at Durham, I had no idea whatsoever of what the acronym "Ph.D." meant. Five years have passed since, and many people have helped me through this long path. With these thanks I intend to reach them all, even those who, for reasons of space, could not be mentioned here.

First of all, I wish to express my gratitude to my partner Grazia and my family, who have been constant and unreplaceable sources of support and encouragement. My most special thanks go also to Maria and Derek, who have guided me through the arduous paths of the research with intuition, enthusiasm and competence.

Secondly, I am grateful to all of my school and university teachers, whose effort has contributed towards my education as a student, researcher and individual. In particular, I would like to thank Prof. Donatiello, for his encouragement and support, and Prof. Zaccarini, for her enjoyable classes and for dedicating me one of her poems.

Finally, I am also grateful to all of my friends, of whom I can name but a few: thanks to Daniel and Alice, for their patience in constantly improving and correcting my English; to Josep, great chess player and table-football mate; to Alex, for not letting our friendship down; to Ali, Pierre and Annamaria, for listening to my argumentations against jealousy; to Romaan, Shehan, Miriam, Edy, Nobuko, Dong-Hoon and all the corridor mates who have made my Ph.D. in Durham a very special one.

Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

Abstract

This thesis is concerned with the problem of construction of the logical *structure* of a persuasive discourse. A persuasive discourse can be defined as a monodirectional form of communication, generated by a speaker in order to convince a hearer about the validity (or fallacy) of a specific belief.

The construction of the structure of a persuasive discourse is realized, in this work, through the adoption of two basic elements: a *belief* system and a *planning* system. The planning system is used as a tool for the automatic generation of the discourse structure (or *plan*), obtained through the decomposition of the assigned (communicative) goals of persuasion, aimed at producing specific effects on the hearer's beliefs. The belief system is adopted in order to endow the planning process with a formal language of beliefs for the representation of such goals, and with the mechanisms which govern the propagation of their (expected) effects on the rest of the hearer's belief state.

The main results presented consist of the formalization of a paradigm for specification of belief systems, and of a method — whose correctness is formally proved — for their *integration* with planning systems. The formalization of a belief system for discourse structure representation (defined in accordance with the theoretical paradigm) is also given, together with the description of its implementation and integration with a specific planner, which resulted in the actual completion of a system for the automatic generation of persuasive discourse plans.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Method

The spirit of this investigation has been guided by the fundamental criterion of *usefulness*: the main goal of this research has been identified with the intention of producing results which could be used for solving actual problems in a specific area; moreover, the contents of the thesis have been restricted to material which is *believed* to be useful for the specific goal which has been chosen.

By taking this 'pragmatist' point of view, the ideas, terms, definitions and results which did not satisfy the criterion of usefulness have been eliminated from the analysis. On the other hand, the elements which form the various parts of the thesis have each been included for a specific reason, which has been made clear in the explanation and which constitutes a contribution to the overall purpose of the work. In accordance with this criterion, whenever an assumption has been made, its adoption has been explicitly stated and motivated.

Nevertheless, the contents of this work still constitute the expression of personal and possibly fallible reasoning. In order to facilitate the process of analysis and validation of the results presented, any claim which has been put forward in the course of the discussion has been justified, either *informally* (e.g. by referring to results obtained by others, in which case the sources have been explicitly stated), or *formally* (i.e. by adopting formal methods for the representation and proof of the claims), or *empirically*, that is, through the analysis of results of practical experiments, or, finally, through any combination of these methods.

1.2 Introducing the problem

Communication in natural language has been and still constitutes one of the most characterizing and complex activities of human beings: regardless of whether it is spoken, written on paper or encoded on electronic support, natural language (NL) represents the primary means adopted by people in order to transfer information from one individual to another.

During the last two decades, the need for the development of a user-friendly information society has produced an increase of interest towards the problem of man-machine communication. The applications of this area of research include interfaces for knowledge bases, information retrieval, text summary, decision support and advisory systems, expert systems and computer aided learning systems interfaces; in fact, any situation in which nontrivial reasoning needs to be communicated to a human user, or vice versa.

Although the largest part of human linguistic exchanges occur as speech, the broad problem of spoken language processing can be tackled, initially, by limiting the analysis to the simpler sub-problem of (unambiguous) *written* language processing. In particular, this work is concerned with the problem of automatic generation of the *structure* of a discourse.

A discourse *structure* consists of an abstract 'skeleton' which captures the *logical* and *intentional* relations existing among the semantic units or *concepts* which will have to be conveyed in the message; the discourse structure does not necessarily contain any explicit NL expression.

The importance of the construction of an underlying 'palimpsest' of the discourse to represent the logical and intentional structure of the message during the phase of generation (and understanding) has been recognized by several authors in the fields of linguistics and computational linguistics (e.g. [63], [98], [74], [153], [121], [99]).

Although a more specific definition of discourse and discourse structure will be given later on in the thesis (Section 2.3), it is useful to point out that one of the main hypotheses at the basis of this work consists of assuming discourse to be the result of a *goal oriented* process: people communicate for a specific reason. This initial reason normally leads the speaker to identify a set of *communicative goals* [98] (or *intentions*) which constitute the effects that (s)he intends to produce on the hearer's mind.

More precisely, this thesis focuses on the class of discourse structures which are

generated by a specific type of communicative goal, namely, that of *persuading* the audience of the validity — or, alternatively, of the fallacy — of a certain proposition or fact. Hence, the class of discourses considered has been restricted to that of *persuasive* discourses.

The decompositional nature of intentions and goals along with the tree-like structure of discourse have led to the adoption, in AI, of the *planning* technique as a widespread approach to the problem of automatic discourse generation and processing (e.g. [25], [5], [64], [100], [92], [74], [98], [153]).

In this work, the same approach has been adopted: the problem of the discourse structure generation is treated as an application domain for the theory of planning, in which the objective consists of the development of systems for the automatic generation of *plans* for persuasive discourses.

However, as mentioned before, the scope of this thesis has been restricted to the problems concerning the generation of the *high-level structure* of a discourse plan, excluding the issues related to the low-level refinement of the message to be communicated, such as the selection of the specific terms in NL, the attention shift or the choice of the linguistic register. Hence, the discourse plans which will be considered will only contain a non-refined representation of the message.

The adoption of planning for the automatic generation of discourse structure also allows the representation, through the plan which is being constructed, of the entire intentional skeleton and logical relations which underly the selection of the discourse contents. Notice that the specific contents to be conveyed by the message will be represented in terms of primitive semantic units (or *events*) inside the current planning 'state', which will contain the set of knowledge and beliefs of the system (that is, of the speaker).

Planning a communication that achieves a specific *change* in the hearer's mental state and attitude towards certain propositions requires the speaker to hypothesize and maintain an adequate *model* of the hearer's beliefs, which must be updated according to the effects that the speaker's utterances (are expected to) have on such beliefs.

As a matter of fact, a generic speaker must always assume a certain minimum level of *knowledge* and *deduction abilities* (rationality) in the audience: these assumptions, taken together, constitute what has been defined as the speaker's model of the hearer. In the theories of discourse and communication, the need for the

speaker to maintain an internal representation of the hearer's knowledge and beliefs has been widely acknowledged (e.g. [25], [98], [145], [10], [11]). Quoting Grosz and Sidner:

> "[...] any model (or theory) of the communication situation must distinguish among beliefs and intentions of different agents". [64, p.425]

In is important to notice that for a discourse processing system involved in an informational exchange, not only is modelling the beliefs and intentions of the other agent(s) needed during the generation phase, but, as Moore and Pollack have underlined in [99], it can be also fruitful during the process of discourse interpretation and contents understanding.

In the computational theories of discourse generation and understanding there exist various examples of formalized languages of beliefs and intentions which have been developed to describe the model of the hearer as a part of the speaker's cognitive state (cf., for example, [10], [12], [153], [92]).

However, one of the characteristics of the languages of beliefs is that the more *expressive* they are, the more *complex* their use becomes. The growth of complexity is due to the high number of *redundant* expressions and *interactions* between different propositions that an expressive language introduces by getting closer and closer to a real natural language.

As a consequence, if these languages are adopted to represent mental states of the speaker during the process of discourse planning, their use must be regulated by specific rules and axioms deduced from the *meaning* of the expressions, in order to avoid the generation of incorrect belief states representations.

In other words, planning in the domain of 'belief worlds' can involve the definition of extremely complex operators, which should take into account the meaning and consequent effects of *propagation* of every belief change of the state, in accordance with the semantics of the specific language defined. In order to avoid this situation, the approach suggested in this work consists of dividing the problem into two distinct parts: first of all, it seems appropriate to identify clearly the language of beliefs, the meaning of the various expressions and the interactions present between them, as well as the restrictions to which a 'correct' belief model must be subject. In brief, this corresponds to giving a complete formal specification of a *belief system.*

Secondly, it will be necessary to effect the *integration* of the belief system previously defined into a planning system, in order to allow the use the of language of

beliefs for the specification of the speaker (and hearer) mental states and planning *operators*, which will be employed to produce the desired discourse plans.

As it will be shown, this approach presents several advantages, from the point of views of the complexity of the development, generality of the solution and maintenance of the final product.

1.3 Statement of the problem

Having traced the broad outline of the contents of the thesis, it is now appropriate to give a clear definition of the problem, in order to delimit the scope of the work, identify the main objectives and introduce the methods for the evaluation of the results.

The central problem tackled in this thesis consists of analysing, formalising and implementing a *belief system* which is intended to be used — in conjunction with a planning system — for the automatic generation of *persuasive discourse plans.*

The requirements that the belief system which will be integrated in the planning system has to satisfy are summarised by the following points:

- the ability to adopt rules of *inference* in order to deduce new beliefs from existent ones, and, at the same time, to limit such process of deduction to the minimum amount necessary, in order to allow the realisation of systems with *limited resources*;
- the ability to deal with *uncertainty* and with *inconsistent* information;
- the ability to represent a structure of belief *justification* and *grounding* based on the formalization of inferential and non-inferential — i.e., *experience*-based — beliefs.

The model of beliefs developed will allow the representation of the beliefs of the system from the point of view of the *speaker*, and will also have to be able to contain a representation of the (hypothetical) model of the hearer.

Notice that the audience will always be considered as a single hearer, even when actually consisting of many. In such cases, the model of the hearer should be regarded as composed by the set of beliefs resulting from the intersection of the belief sets of all the members of the audience.

The evaluation of the model of beliefs developed will be based on two parameters which result to be critical for most of the formal logics and reasoning systems, and which are, in general, antagonist: *expressiveness* and *efficiency*. Moreover, the formal *correctness* of the solution proposed will also constitute a necessary requirement.

Being the belief system designed to be used by a planning system to produce plans for persuasive discourses, these aspects will also be evaluated with respect to the problem of persuasive discourse planning itself. This means that, in order to be expressive, the language of beliefs will have to be sophisticated enough to capture the *relevant information* and concepts which will be needed during the process of construction of the logical structure of a discourse.

Similarly, the efficiency will be evaluated as the workload necessary to *integrate* the belief system into a planning system, plus the impact which the result of the process of integration itself has on the efficiency of the planner. In addition, two other aspects of the integration will also be considered as parameters of efficiency of the solution proposed: the first one consists of the range of different planning systems into which the belief model developed will be able to be integrated; the second one corresponds to the complexity of the possible procedure of maintenance of the final product.

Finally, the requirement of correctness of the solution will be considered satisfied only if the formal specifications of the belief system developed are fully met by the model actually implemented, and after a *formal proof* of the validity of the process of integration has been given. A more specific description of these criteria of evaluation will be given in Chapter 2, Section 2.5.

In order to complete the definition of the characteristics and scope of the problem, it is necessary to introduce one of the main hypotheses which will be assumed throughout the thesis, concerning the expected 'attitudes' of the hearer (and speaker) towards communication in general.

With regard to the general principles which regulate conversational exchanges, Grice (in [61]) introduced the maxim quoted below, defined as 'the supermaxim of Quality', which constitutes part of the general principle of *cooperative conversation*:

> "Try to make your contribution one that is true." [61, p.46]

In what follows, persuasive discourse, even though aimed at affecting the hearer's beliefs concerning certain propositions, will be assumed to be totally *sincere*. More

specifically, Grice adds the two following principles:

1. Do not say what you believe to be false;
2. Do not say that for which you lack adequate evidence.

In the model developed in this thesis, these two maxims are assumed to hold during the complete process of discourse planning. This implies that the speaker's strategy of persuasion cannot make use of deceptive techniques, rhetorical tricks or fallacies. What is more, the hearer will be expected to be aware of this 'sincere approach'. In other words, the two basic assumptions upon which the discourse construction will rely are as follows:

Axiom I) *The speaker is* sincere;
Axiom II) *The hearer* believes *the speaker to be* sincere.

From now on, these two principles will be referred to as *sincerity axioms*. It should be noticed that these assumptions are not enough to guarantee that the hearer will *believe* (or be persuaded of) anything the speaker says. Sincerity, by itself, is not sufficient to convince: after the discourse, the hearer could still maintain that the speaker's opinions are completely mistaken, even though totally honest.

1.4 Main contributions of the thesis

The main contributions of this work lie in the field of discourse planning and consist of:

1. the *formal* specification of a 'speaker-hearer' belief system for discourse planning (Chapter 4);
2. a sound linear algorithm for the integration of the speaker-hearer belief system into a planning system (Chapter 5).

The innovative aspects of the belief system developed are essentially its ability to offer all the basic features required to an expressive resource-bounded speaker-hearer belief system (as stated in the previous section) and its suitability for the *automatic integration* into an entire *class* of planners.

The main advantages of the integration algorithm adopted, consisting of the pre-processing of the rules which govern the belief system into the *planning problem* itself, lie in its formal correctness and linear complexity. Moreover, as it will be shown in Chapter 5, the algorithm produces planning problems which are formally correct and can be solved more efficiently than those produced by other methods of pre-processing.

In conclusion, not only will the results presented in this work satisfy the declared goals of the thesis, but constitute also a useful contribution to the current research in this field, which will be surveyed in the next chapter.

1.5 The structure of the thesis

The content of this thesis has been divided into seven chapters. Following this introductory chapter, Chapter 2 contains the review of the current research related to the problem considered, and a more precise definition of the evaluation criteria.

Chapter 3 introduces the formalization of a *paradigm* for the specification of belief systems, not necessarily tailored to the problem of discourse generation. This chapter defines the high-level structure, functioning and characteristics of a general model, according to which the system actually implemented has been designed.

Chapter 4 contains the detailed, low-level description of the belief system for persuasive discourse planning actually implemented, built according to the theoretical paradigm described in the preceding chapter. The chapter includes the formal specification of the system and a brief description of its realisation.

The main task of Chapter 5 consists of presenting a method for the automatic integration of a belief system into a generic planning system, and of demonstrating its *correctness*. The set of models of beliefs considered for integration contains all the systems which can be built according to the formal paradigm described in Chapter 3. On the basis of a theoretical analysis, these systems are shown to be generally integrable into a wide range of planning systems through a procedure which presents linear computational complexity. This result is formally proved for part of the set of belief systems considered, the proof of the correctness of the integration algorithm relying on a specific set of assumptions. Nevertheless, the belief system presented in Chapter 4 does satisfy such assumptions.

Chapter 6 contains the description of the realisation of a discourse planning system, obtained as the result of the integration between a specific belief system

(consisting of a simplified version of the system of Chapter 4 still satisfying the necessary requirements for the integration) and a specific planning system. In the final part of this last chapter, some examples of belief situations and resulting plans generated for the achievement of *persuasive* communicative goals are illustrated.

Finally, Chapter 7 evaluates the results of the work, according to the criteria of evaluation which will be established at the end of Chapter 2.

Chapter 2

Related Work

2.1 Introduction

The subject of this thesis overlaps mainly with three different trends of research. These three main areas are:

1. belief systems;
2. planning;
3. theories of discourse and argument structure.

This chapter dedicates one section to each of the mentioned areas, in which the relevant works are reviewed. The subject of belief system is presented first and is treated in more details than the others because of its pre-eminent position with regard to the declared aim of the thesis. The section on planning is the last presented, and represents, to some extent, a '*traît d'union*' between the previous two.

Finally, the fourth section restates more accurately the criteria of evaluation which this work will adopt, taking into account the literature surveyed.

2.2 Belief systems

The history of the evolution of the human thought is driven, at least, by two fundamental 'urges', which are often in competition.

The first of these forces consists of the need of the human beings to develop, in the course of their lives, a set of basic *truths*, principles, or 'datum points', upon

which to rely for undertaking action.

The second one consists of the natural inclination of the human thought to the process of rational *enquiry*.

Since ancient times, innumerable systems of truths have been developed in philosophy, theology and science. However, the application of rational questioning has revealed their points of weakness, their inconsistency and incompleteness, producing a continuous, historical process of construction of new '*frameworks*' of beliefs, each built in order to replace the previous, obsolete model (see [86]).

Historically, the use of rationality has not been shown to be incompatible with the development of 'religious' systems of beliefs. One of the most important examples, in the western civilization, is represented by the work of St Thomas Aquinas (1225–1274), who, in his *Summa Theologica*, tried to give a "...complete description of the relation between man and God, relying only on philosophical reasoning, and without recourse to mystical assertion or unsupported faith. [...] [T]he subsequent synthesis of Christian doctrine and Aristotelian metaphysics [...] has remained to this day the most persuasive of the foundations offered for Christian theology" [129, p.17].

Another interesting example can be found in the '*ontological proof*' for the existence of God. This 'proof', normally credited to St Anselm, Archibishop of Canterbury (1033–1109), represents one of the arguments at the basis of the medieval theology, and makes use of one of the main strategies adopted in formal logic for the proof of theorems, viz., *reductio ad absurdum* (see [129, p.21]).

The rise of the modern philosophical thought and scientific investigation is marked by the publication of the work of René Descartes (1596–1650). In his writings, Descartes openly declares that all previous results of philosophy and speculation were without foundation and had to be set aside or suspended until clear and indisputable *premisses* could be established, together with a *method* and principles whereby to advance from them.

In his search for the truth, Descartes was guided by his famous 'method of doubt', which consisted of disregarding any assertion or belief which could be considered dubitable. In Descartes' view, everything could, in principle, be doubted: not only the evidence of the senses and the evidence of memory, but even the most basic presuppositions of scientific thought.

Hence, as indubitable basis for his system of beliefs, Descartes put a proposition which he considered as 'self-verifying': "I think". According to Descartes, this proposition is peculiar in that it cannot be entertained without at the same time

and for that very reason being true. This truth is known, as Descartes puts it, by the 'natural light' of reason, that is, by a process which can be perceived to be valid by anyone who reasons at all.

An interesting point about Descartes's premiss consists of the fact that its validity is *contingent*: the accent is put on the process of the *subject* perceiving the truth of the proposition, and not on its immutable necessity. In other words, the proposition should be regarded more as a *belief*, rather than as an objective *truth*.

During the first half of this century, the scientific world has undergone another major crisis, brought about by the discovery of the existence of paradoxes in the set theory, of non-Euclidean geometries and of the Trans-finite set theory. This crisis culminated with the publication of Gödel's theorem of incompleteness, which put an upper limit to the power of general validity of any consistent formal system complex enough to contain the basic principles of the arithmetic (for a discussion on the relation between Gödel's theorem and AI, see [58, pp.113–156]).

One of the consequences of this revolution consisted of the rise of a current of thought known as *falsificationism*, mainly represented by the work of Karl Popper (see [115]).

According to Popper, any theory of knowledge is destined to be proved, sooner or later, incomplete. That is, for any given theory, there are always 'exceptional' cases in which the expected results will not coincide with the results of the real experience. Therefore, in order to include these 'anomalies' in the model, a new, more complete theory will have to be constructed. But such theory, in turn, will be shown wrong in some cases, and so on, *ad infinitum*.

As a result, the idea that the human mind is able to acquire objective *knowledge* is undermined at its very foundations, and is replaced by the 'weaker' concept of temporary — and, thus, subjective — systems of *beliefs*.

The development of theories of beliefs and knowledge during the last decades can be divided, roughly, into two currents: the first one, mainly deriving from the field of philosophy, has produced rather *informal* works. The second one, emerging from a branch of classical logic, has been marked by the construction of numerous *formal* theories. The next section will examine some interesting examples of the former kind of approach, while the more formal theories will be considered in Section 2.2.2.

2.2.1 Informal approaches

This section analyses three informal works on belief systems, namely, those of Ackermann [1], Armstrong [8] and Harmann [66]. These examples have been chosen because, taken together, they tackle most of the important issues related to the subject; moreover, their point of views are, to some extent, complementary. Furthermore, although the first two examples were published early in the '70s, their ideas can still be found in much later works on belief systems, including formal theories — which will be discussed in Section 2.2.2 — and more recent informal approaches, as the analysis of Harman's work will show.

The work of Ackermann

A representative example of informal analysis of belief systems can be found in the work of Robert Ackermann, '*Belief and Knowledge*' [1].

Ackermann begins by dividing the actual beliefs of a human mind into three categories: *behavioural* beliefs, *unconscious* beliefs and *conscious* beliefs. Into the first category fall those 'assumptions' which rational agents adopt — without explicitly formulating them at a conscious level — when undertaking actions in daily life. For example: Mary, intending to have Cheerios for breakfast, goes to the cupboard and opens it. When performing such an act, she relies on the (unstated) belief that the Cheerios *are* in the cupboard.

To the second category belong "long-standing beliefs that can influence behavior over a long period of time, but which resist recognition by the agent."([1, p.6]). The main example of this group consists of the set of *prejudices* which can be deeply and unconsciously act in somebody's mind.

Finally, any belief that a person has explicitly formulated and which is aware of falls into the class of conscious beliefs.

Besides these three categories of beliefs, Ackermann also introduces a fourth type of beliefs, which he calls *rational*, representing "a philosophical idealisation of actual belief structures as they are found in human beings" ([*ibid.*, p.8]). Ackermann then imposes that a particular set of rational beliefs be *consistent* (i.e. do not contain contradictions) and *complete*, "in the sense that all of the logical consequences of a given consistent set of beliefs should be regarded as belonging to a rational set of beliefs."([*ibid.*, p.9]). However, he points out immediately afterwards that the "completeness condition is strong since it plainly requires that a man having any belief also have as beliefs the *infinite* number of consequent beliefs

that can be drawn from that belief by valid inference. The *completeness condition* reveals the essentially ideal nature of rational belief."

This problem, well known in formal logic, is commonly referred to as the problem of *logical omniscience* of a rational agent (a term introduced by Hintikka in [69]). With regard to this issue, the view which Ackermann puts forward is contained in the following quotation:

> "We will take a person's actual beliefs to be rational if they could be taken as a *proper subset* of the set of beliefs of some ideally rational agent whose rational belief structure satisfied the conditions of consistency and completeness [...]. An even weaker sense of rationality is provided if we say a man is rational should he always be willing to *revise* his beliefs when it is demonstrated that they are not incorporable in the belief structure of an ideally rational agent." [*ibid.*, pp.9–10] (The italic is mine).

The interesting point to notice is that, according to Ackermann, although a real agent has to be ready to *accept* — once put in front of the rational evidence — *any* of the logical consequences of his beliefs (and, consequently, be ready to revise the set of beliefs to eliminate possible contradictions), (s)he does not necessarily have to be *aware* of *all* of them.

The same idea of 'awareness' is expressed in formal terms by Fagin and Halpern in [37], where the authors extend Levesque's logic ([89]) of implicit and explicit belief to allow multiple agents and higher-level belief (i.e., beliefs about beliefs). Such logic will be reconsidered in more details in the Section 2.2.2.

Ackermann also introduces a syntax for the 'well formed formulae' (*wff*) of the language of beliefs; some of the expressions he uses are listed below:

- Ba (a believes that).
- $\sim Ba$ (a doesn't believe that).
- $BaBa$ (a believes that a believes that).
- $Ba \sim Ba$ (a believes that a doesn't believe that).

Notice that in Ackermann's formalism, a's belief is not iterated more than twice. To justify this restriction, he points out that it is "difficult to imagine circumstances in which 'I believe that I believe that I believe that p' can vary in truth or significance from 'I believe that I believe that p.' " With regard to this issue, Ackermann seem to have adopted, for the notion of belief, the same position that many others hold

with respect to the problem of the distinction between the concept of '*knowing*' and '*knowing to know*'. For example, Schopenhauer writes:

> "If your knowing and your knowing that you know are two different things, just try to separate them, and first to know without knowing that you know, then to know that you know, without this knowledge being at the same time knowing." [68, p.166]

Nevertheless, the validity of this concept is limited to situations which do not involve the nesting of attitudes of different agents.

After the definition of the syntax of the sentential formulae of belief, Ackermann introduces a simple algorithm to check for the consistency of any given set of beliefs. It is interesting to notice that his system allows the possibility for an agent to be 'agnostic', that is, to have an attitude of *uncertainty* towards a specific proposition. In fact, the set of beliefs

$$B = \{ \sim Bap, \ \sim Ba \sim p \}$$

is shown to be perfectly consistent by his checking algorithm ([1, p.27]). In other words, the analogue of the logical law of the *excluded middle* ($p \vee \sim p$) in the model of beliefs of the agent 'a' is not valid.

With regard to the adoption of *probabilities values* to determine more precisely the boundaries between the three qualitative attitudes 'belief', 'disbelief' or 'agnosticism', Ackermann uses the famous example of the 'lottery paradox' ([87]) to show why this strategy does not always produce good results.

Finally, another important aspect concerning Ackermann's view of a rational belief system lies in the need of a structure of belief *justification*, or, as he puts it, of supporting 'evidence':

> "If my beliefs are to be regarded as rational, not only must they be logically consistent, they must also be related to the evidence I have at my command in such a fashion that beliefs which are rendered highly unlikely on that evidence are not believed by me." [1, p.33]

Although Ackermann does not present a specific syntax to express this evidence, such an idea constitutes an important part of his model, especially in relation to the analysis of the concept of *knowledge*. In fact, he (among many others) considers the assertion "knowing that p" as a special case of "believing that p", in which the

proposition asserted must be supported by evidence so strong that it rules out all relevant objections to p:

> "...: a person knows p because he has sufficient evidence to rule out all possible relevant non-metaphysical objections [...]" [*ibid.*, p.75]

However, the notion of 'sufficient evidence' (or 'complete justification', introduced later on) is not clearly defined. As Ackermann points out, this concept is highly dependent on the context: for any particular knowledge claim, one must examine the circumstances to determine whether there are objections and whether they can be met in order to determine whether the knowledge claim is true.

In conclusion, Ackermann's analysis seems to suggest that the approach to the representation of human knowledge should be based on a system of beliefs, and should allow the adoption of a structure to represent the evidence supporting any specific claim. Moreover, being such evidence not clearly definable in objective and generally applicable terms, it seems that the justification structure will be doomed to be identified on the basis of *subjective* and context-dependent criteria.

The work of Armstrong

A second interesting informal account on belief systems and knowledge is given by Armstrong in [8]. This work, although contemporary with Ackermann's, is presented with a less 'rational' slant, and seems to be more concerned with the philosophical issues related to the matter.

To begin with, Armstrong sees beliefs as 'maps' by which we 'steer', that is, entities upon which we rely when undertaking action: "*beliefs are, mere thoughts are not, premisses in our practical reasoning.*"[Op.cit.,p.74]. The analogy with maps is also used to describe the idea of *introspection*: using Armstrong's words, the "belief-map will include a map of the believer's own mind, and even, as sub-part of this sub-part, a map of the believer's belief-map (that is, his beliefs that he holds certain beliefs)." However, according to his view, this entails no vicious infinite regress, because the "belief-map is not a complete map of the world, and [...] the map of itself that it contains is even more incomplete [...]."([*ibid.*, p.4]).

In order to give a more precise definition of what a belief is, Armstrong describes beliefs as 'states': according to his view, an agent's belief in p is a matter of the agent's being in a certain 'continuing state', a state which endures for the whole time that (s)he holds the belief. Interesting enough, this interpretation of beliefs as states leads him to allow for the existence of 'irrational' belief systems:

> "[...] [A] man who recognizes the simultaneous existence of logically incompatible beliefs in his own mind must also recognize that his cognitive state is irrational. But recognition of one's own irrationality does not necessarily abolish it. And if beliefs are *distinct structured states*, then it is easy to see how the belief that p, the belief that $\sim p$, and the knowledge both that the two beliefs are held and that they are incompatible, could co-exist in the one mind." [*ibid.*, pp.105–106]

According to Armstrong, this view is supported by the psychological hypothesis of the presence, in the human mind, of multiple belief systems which exist in relative isolation from each other. This idea has been also advocated by other philosophers; for example, Robert Stalnaker writes:

> "A person may be disposed, in one kind of context [...] to behave in ways that are correctly explained by one belief state, and at the same time be disposed in another kind of context [...] to behave in ways that would be explained by a different belief state." ([139, p.83])

In order to sustain the possibility of the co-existence of contradictory beliefs, Armstrong proposes also a more 'pragmatic' point of view:

> "[...] [T]he human mind is a large place and untidy place, and we may believe 'p' and '$\sim$p' simultaneously but *fail to bring the two beliefs together*, perhaps for emotional reasons. [...] In the above sorts of case, the believer has two beliefs which are very obviously logically incompatible, but, because he does not bring them together, is not *aware* of holding incompatible beliefs." [*ibid.*, pp.104–105] (The italic is mine).

Thus, as in Ackermann before, there is the idea that a — resource limited — rational agent could be not fully 'aware' of *all* of the consequences of the beliefs (s)he holds, including, in this case, possible inconsistencies. In order to solve such situations, it would seem necessary to draw the agent's 'attention' simultaneously on the two contradictory beliefs.

It is appropriate, at this point, to make a brief digression. With regard to the existence of contradictory beliefs, it should be noticed that, amongst the theories of beliefs based on a formal approach, there is a family of *non-Cartesian*[1] logics —

[1]The non-Cartesian logics are those in which there are no theorems of the form '$Believes_a P$', implying, in some sense, the existence of propositions which must be believed by everyone.

namely, *paraconsistent* modal logics — which are *inconsistency-tolerant.*

These logics are characterized by the absence of the logic theorem

$$(P \wedge \neg P) \rightarrow Q$$

known as *ex falso quodlibet* (anything can follow from contradiction). The problem with this theorem lies in the fact that, when a contradiction is deduced from a set of beliefs, it also follows by *ex falso* that *any proposition*, including every possible contradiction, follows validly from that set of beliefs. Therefore, if we apply a classical inference engine to drawing conclusions from an inconsistent set of beliefs, the inconsistency spreads across every part of the system.

However, this does not seem to be what happens in real-life belief systems: reasonable people normally *isolate* inconsistency, or temporary suspend judgement, but certainly do not *exploit* contradiction to deduce the validity of any belief they happen to need to prove. As a consequence of the fact that also artificial agents have, after all, limited resources, there has been a recent resurgence of interest in these logics, which seem to be able to deal with contradiction in a more rational fashion than that indicated by classical ones (see, for example, [59]).

It is interesting to examine briefly Armstrong's categorization of the rules of inference adopted by an agent to deduce new beliefs from the existent ones. Such rules are classified as 'general beliefs', by which Armstrong means 'dispositions' that the believer has to *extend* the set of beliefs. There are two interesting aspects related to his definition of such beliefs: first of all, they are not considered 'part of the map', i.e., they are not actually regarded as 'beliefs'; secondly, their validity is totally *subjective*:

> "So, I suggest, if 'q' is to be (one of) A's reason(s) for believing that 'p', there must be some *general principle* operating in A's mind according to which it is possible for A to move from the first proposition to the second. The general principle *need not be true, nor have any plausibility to anybody but A*. Nor need A be aware that the principle operates in his mind." [*ibid.*, p.85] (The italic is mine).

Armstrong seems to consider the process of *inferring* new beliefs as one of belief grounding: according to his view, "a proposition '*q*' is a *sufficient ground* for '*p*' if, and only if, the belief states Baq and Bap are *causally* connected"[2] [Op.cit.,p.93],

[2] 'Bap' should be read as "Agent 'a' believes p".

that is, if the first belief-state brings the second belief-state into existence and acts as *sustaining* evidence for it. Later on, Armstrong also discusses the concept of 'reason' for justifying a belief:

> "[...] [T]he goodness or badness of a reason has nothing essential to do with its actually *operating for somebody* as a reason. [...] 'q' is a conclusive reason for believing that 'p' if, and only if, it is the case that 'if q, then p'. [...] The proposition 'if q, then p' need not be true of logical necessity. [...]. All that cannot be allowed is reading 'if q, then p' simply as the logician's 'q $\supset$ p'. And a logical link does not make 'q' a more conclusive reason that in the case of a non-logical link." [*ibid.*, pp.96–97]

The only theme which all these considerations seem to have in common is that of the *subjectivity* of the mechanisms of inferencing and belief justification. This finds further support in the fact that, although rejecting the (Cartesian) notion of existence of beliefs which are self-evident, indubitable or incorrigible ("I think the logical possibility of error is always present in any belief about any subject matter whatsoever" [Op.cit., p.156]), Armstrong clearly accepts the existence of *non-inferential* beliefs, which can be held without any need for a specific justification:

> "It is perfectly possible to hold a belief without the holder having any reason for it. Such a belief will be called a 'non-inferential' belief. (That a belief is non-inferential does not entail that it is unreasonable or irrational. [...])" ([*ibid.*, p.77])

The issue of non-inferential beliefs is raised again subsequently, when the relation between knowledge and belief is examined. With regards to this problem, Armstrong adopts the 'classical' view, shared by Ackermann, according to which "...knowledge is a true belief for which the believer has sufficient evidence [...]." [Op.cit., p.152]. However, unlike Ackermann, Armstrong believes that the evidence for knowledge will have to be "...some proposition, 'q,', which is *known* to [the agent] [...]." Hence, knowledge that 'p' will be analysed in terms of knowledge that 'q', knowledge that 'q' in terms of knowledge that 'r', and so on, it would seem, *ad infinitum.*

In order to solve the problem of 'infinite regress' — already known at the times of Plato — Armstrong appeals to non-inferential beliefs, which, according to his

view, consist of "empirically reliable belief[s]" [Op.cit., p.159]. More precisely, he writes:

> "I suggest that at least one place where non-inferential knowledge is to be found is in *the simpler judgements of perception.* [...] I have in mind such judgements as 'There is a noise within earshot', 'It is getting hotter','There is something red and round over there', 'There is something pressing on my body' and so on." [*ibid.*, p.163]

Armstrong describes also what he calls the 'pessimistic' view of non-inferential knowledge. In this view, the actual sensory beliefs should be restated as 'It sounds to me as if there is a noise within earshot', 'It feels to me as if it is getting hotter', and so on; in other words, the accent is put on the *subjectivity* of the physical perceptions.

Summarizing, the analysis of Armstrong draws the attention upon various aspects related to the issue of belief systems, such as the co-existence of contradictory beliefs, the nature of inference rules and the problem of the infinite regress of belief (or knowledge) justification; the main 'ingredients' of the solutions which he proposes for these questions can be condensed in the three following ideas: *subjectivity* of the point of view of the believer, *limited* resources of the rational agent and importance of the *sensorial* experience as a way of belief grounding.

The work of Harman

Several elements present in the works of Ackermann and Armstrong can be found in many other informal approaches which have been developed subsequently. An interesting example is represented by the work of Gilbert Harman ([66]), who, although mainly concerned with the principles of reasoning, proposes an interesting analysis of the problem of belief systems and related issues.

One of the ideas introduced by Harman as a consequence of the consideration of the limited 'storage capability' for beliefs of an agent consists of the assumption that beliefs can be divided into two classes, namely, *explicit* and *implicit* beliefs. Harman defines as 'explicit' any belief which involves an explicit mental representation; on the other hand, "something is believed only implicitly if it is not explicitly believed but, for example, is easily inferable from one's explicit beliefs." ([66, p.13]).

To quote the example that Harman gives, if one explicitly believes the earth has exactly one sun, one can easily infer that the earth does not have *two* suns, that the earth does not have *three* suns, and so on. Hence, although one can have

only a finite number of explicit beliefs, one can implicitly believe *infinitely* many things — most of which one is not *aware* of.

By adopting this view, Harman claims that what he calls the 'Logical Closure Principle' — stating that one agent's beliefs should be closed under logical implication — should be abandoned:

> "Clearly this principle does not apply to explicit beliefs, since one has only a finite number of explicit beliefs, and they [would] have infinitely many logical consequences. Nor can the Logical Closure Principle be satisfied even by one's implicit beliefs. One cannot be expected even implicitly to believe a logical consequence of one's beliefs if a complex proof would be needed to see the implication." ([*ibid.*, p.14]).

This point of view clearly reflects that of Ackermann, who, in relation to the problem of *logical omniscience*, considered the actual beliefs of a person to be rational if they could be taken as a *proper subset* of the set of beliefs of some ideally rational agent (see page 14).

Similar considerations to those presented by Armstrong on the existence of contradictory beliefs in a rational agent can also be found in Harman's work:

> "[...] [S]ometimes one discovers one's views are inconsistent and does not know how to revise them in order to avoid inconsistency without great cost. In this case the best response may be to keep the inconsistency and try to avoid inferences that exploit it. This happens in everyday life whenever one simply does not have time to figure out what to do about a discovered inconsistency." [*ibid.*, p.15]

The analogous of Ackermann's definition of 'knowledge' as true belief supported by evidence which meets *any possible objection* is given by Harman in more pragmatic terms:

> "In fully accepting P, one takes oneself to *know* that P is true. [...] [O]ne is *justified* in fully accepting P only if one is justified in ending one's investigation into whether P is true. This means one has to be justified in implicitly supposing that further investigation would not be sufficiently worthwhile, for example, by uncovering relevant evidence of a sort not yet considered. [...] A hypothesis is 'corroborated' only to the extent that it survives one's best attempts to refute it." [*ibid.*, pp.47–48]

It is interesting to notice that Harman solves Ackermann's problem of defining precisely the meaning of 'any possible objection' by taking — as anticipated — a *subjective* position.

In relation to the problem of the regress of belief justification, Harman presents two theories, the *foundations* theory and the *coherence* theory. The former theory holds that "some of one's beliefs 'depend on' others for their current justification; these other beliefs may depend on still others, until one gets to foundational beliefs that do not depend on any further beliefs for their justification." [*ibid.*, p.29].

On the other hand, according to his definition of coherence, one's ongoing beliefs ought not to have the justificational structure required by the foundations theory: "Justification is taken to be required only if one has a special reason to doubt a particular belief." [Op.cit., p.29].

Although Harman clearly supports the view that people do not keep track of the justification relations among their beliefs because, for a resource limited agent, this "would involve too much record keeping" (p.115), he still suggests that, under the coherence theory, the following basic principle should be adopted for belief revision:

> "*Principle of Positive Undermining* One should stop believing P whenever one positively believes one's reasons for believing P are no good." [*ibid.*, p.39]

Clearly, the idea of keeping track of the 'reasons' for believing a specific belief is still present.

Finally, it should be pointed out that, along the lines of a more 'resource sparing' theory, Harman introduces also the idea that belief revision should involve *minimal changes* in one's beliefs, and should be adopted exclusively when it increases the overall coherence of the belief system:

> "The coherence theory supposes one's present beliefs are justified just as they are in the absence of special reasons to change them, where changes are allowed only to the extent that they yield sufficient increases in coherence." [Op.cit., p.32]

These concepts have been subsequently adopted for the development of formal models of belief system revision (e.g. [54], [3]). Such models will be examined, to some extent, in the following section.

2.2.2 Formal approaches

The various formal approaches to the problem of belief systems can be classified in terms of two parameters: the model of belief adopted, which determines the *semantic* characteristics of the system, and the language used to formalise the model, which specifies the *syntax* of the expressions which the system will be able to deal with (cf. also [151]).

With regards to the model adopted, there are two basic approaches: the first, best-known, and probably most widely used solution consists of adopting a *possible worlds* semantics, where an agent's beliefs are characterized as a set of so-called 'possible worlds'. In such semantics, belief is taken to be a *relation* between the agent and abstract propositions about the world: the model attributes specific propositional attitudes (knowledge or belief) to an agent, but does not impose any *cognitive structure.*

The most common alternative to the possible worlds approach consists of adopting a *sentential*, or *interpreted symbolic structures* model. In this kind of models, an agent's beliefs are characterized by the computations an agent performs on syntactic objects (symbols or sentences) explicitly represented in the data base of the agent; an epistemic state is normally represented as a *set of propositions*, and an agent believes a proposition ϕ iff ϕ is present in the belief set.

With respect to the language of beliefs, there are, traditionally, two fundamental approaches. The first consists of using a *modal* language, which contains non-truth-functional *modal operators* applied to formulae. The second, so called *syntactic* approach, involves the use of first-order (truth-functional) 'metalanguages', and is normally adopted for sentential models. The differences between these languages can be found both on a syntactic and semantic level.

From a semantic point of view, in classical (propositional or first-order) logic, the truth value of an expression is dependent solely on the truth values (*denotations*) of its sub-expressions. For example, the denotation of the propositional logic formula $(p \wedge q)$ is a function of the truth-values of 'p' and 'q'. However, the notion of belief does not seem to be truth functional: as noticed by Wooldridge and Jennings in [151], "It is surely not the case that the truth value of the sentence '*Janine believes that* p' is dependent solely on the truth-value of p [...]."

From a syntactic point of view, the argument 'p' of a modal expression $B_a p$ is normally considered as a proposition of the language of beliefs itself. In contrast, the argument 'e' of a predicative formula $Bel(a, e)$ is taken to refer to an expression

in some 'object' language, whereas the language of *Bel* constitutes a metalanguage. Quoting Konolige:

> "In the so-called *syntactic* approach, the language is a first-order metalanguage that contains terms whose denotations are expressions in an object language. The object language serves as the internal language of the belief system, and predicates in the metalanguage express facts about these sentences, such as their inclusion in the base set of sentences." [82, p.6]

Notice that if the metalanguage and its object language coincide (i.e. '*e*' can refer to expressions in the language of *Bel*) the metalanguage is said to be *self-referential*; otherwise, it is *hierarchical*.

Although various examples of first-order metalanguages have been developed (e.g. by Konolige [81], Haas [65], Morgenstern [101] and Perlis [109] [110]), the main weak points of these formalisms seem to be their predisposition to fall prey to inconsistency (cf. [141]) and their notational complexity, which "makes them very hard to work with." [82, p.84] (cf. also [150, pp.28–32]).

The basic possible worlds model and the other alternative models of beliefs which will be analysed in the subsections that follow adopt, to specify the syntax of the language of beliefs, the 'modal' approach.

The possible worlds model

The possible worlds model for logics of knowledge and belief was originally proposed by Hintikka [69], and is now most commonly formulated in a normal modal logic using the techniques developed by Kripke [85].

A normal modal logic is essentially a classical propositional logic, extended by the addition of two operators: '$\Box$' (necessarily) and '$\Diamond$' (possibly). Let $Prop = \{p, q, \dots\}$ be a countable set of atomic propositions. Then, the syntax of the logic is defined by the following rules: (i) if $p \in Prop$ then p is a formula; (ii) if ϕ, ψ are formulae, then so are $\neg\phi$ and $(\phi \vee \psi)$; and (iii) if ϕ is a formula, then so are $\Box\phi$ and $\Diamond\phi$. The operators '$\neg$' (not) and '$\vee$' (or) have their standard meanings. The remaining connectives of classical propositional logic con be defined as abbreviations in the usual way.

The formula $\Box\phi$ is read "necessarily ϕ"; analogously, $\Diamond\phi$ is read "possibly ϕ". The semantics of the modal operators are given by introducing 1) an *accessibility relation* R, which defines what worlds are considered accessible from the current

world, and 2) a *valuation function*, which says, for each world w, which atomic propositions are true in w. The formula $\Box\psi$ is then true if ψ is true in every world accessible from the current world; $\Diamond\psi$ is true if ψ is true in at least one accessible world.

The two modal operators are *duals* of each other, i.e., each can be defined in terms of the other:

$$\Box\psi \Leftrightarrow \neg \Diamond \psi$$

A formula ψ is said to be *valid* if it is true in all the possible models, where a 'model' simply consists of a specific collection of possible worlds, of an associated accessibility relation and a valuation function. To express that ψ is valid, the standard notation '$\models \psi$' is normally adopted.

The two basic properties of this logic are the following:

K) $\models \Box(\phi \Rightarrow \psi) \Rightarrow (\Box\phi \Rightarrow \Box\psi)$;

N) If $\models \phi$ then $\models \Box\phi$.

The first axiom is called K, in honour of Kripke; since it is a valid formula, it will be contained in any normal modal logic. Similarly, the second property, generally called the *necessitation rule*, will appear as an inference rule of any complete axiomatisation of normal modal logic.

The most interesting properties of normal modal logics derive from the characteristics of the accessibility relation R, defined for each model. For example, consider the following axiom schema: $\Box\phi \Rightarrow \phi$. It turns out that this axiom is true in all and only the models which have a *reflexive* accessibility relation.

The study of the way that properties of R correspond to axioms is called *correspondence theory*. The four main axioms which are associated to important properties of R and which are the basic 'building blocks' of any normal modal logic are reported below:

T $\Box\phi \Rightarrow \phi$	D $\Box\phi \Rightarrow \Diamond\phi$
4 $\Box\phi \Rightarrow \Box\Box\phi$	5 $\Diamond\phi \Rightarrow \Box \Diamond \phi$

Through the correspondence theory, it is possible to derive completeness results for a range of simple normal modal logics. These results provide a useful point of comparison for normal modal logics, and make the possible worlds approach an

attractive mathematical tool to work with [21].

The logics of knowledge (*epistemic* logics) and the logics of belief (*doxastic* logics) are traditionally derived from modal logics by interpreting the formula $\Box\psi$, respectively, as "Agent *knows* that ψ" and "Agent *believes* that ψ". Let us consider the meanings of the basic axioms T, D, 4 and 5 under the doxastic interpretation.

Axiom D says that an agent's beliefs are non contradictory. In fact, using the dual definition of the modal operators, it can be rewritten as: $B_a\psi \Rightarrow \neg B_a\neg\psi$.

Axiom T says that what the agent believes is true. This axiom is only acceptable adopting a *subjective* interpretation of the notion of belief system: if one believes a proposition, one considers that proposition to be true, regardless of whether it is actually true in the real world.

Axiom 4 is called the *positive introspection* axiom, and implies that an agent believes to believe a believed proposition, recursively. Axiom 5, similarly, is called the *negative introspection* axiom, and says that an agent is aware of what (s)he does not believe.

It should be pointed out that, given the respective epistemic and doxastic interpretations of the modal operators, the axioms KD45 (identifying the so called 'weak-S5' system) are often chosen as logic of (idealised) belief, whereas KTD45 as a logic of (idealised) knowledge (cf. [151]).

Nevertheless, under this interpretation, the two fundamental properties K and N of the normal modal logics turn out to be rather problematic.

In fact, according to the axiom K, the knowledge (or beliefs) of an agent is *closed under implication.* Together with the necessitation rule, this axiom implies that an agent's beliefs are closed under logical consequence: an agent must believe all the logical consequences of the beliefs (s)he is currently holding.

This property, already mentioned in the Section 2.2.1 as the *logical omniscience* problem, together with the problem of the widespreading of inconsistency (an agent containing one pair of contradictory beliefs is allowed to believe everything), seem to make the possible world approach unsuitable for representing resource bounded believers. As Levesque points out:

> "Any one of these [problems] might cause one to reject a possible-world formalization as unintuitive at best and completely unrealistic at worst." [89]

As a result, many researchers have attempted to develop alternative formalisms for representing belief. In the subsections that follow, some of these attempts are examined.

Implicit and explicit belief

The concept of differentiating between implicit and explicit beliefs, already found in the work of Harman (see Section 2.2.1), is also at the basis of the formal work developed in [89] by Levesque.

As in Harman, the intuitive idea is that a resource bounded rational agent has a relatively small set of explicit beliefs, and a very much larger (in fact, infinite) set of implicit beliefs, which includes all the logical consequences of the explicit beliefs.

To define the semantics of implicit and explicit beliefs, Levesque proposes the use of *situations*, a notion borrowed from situation semantics (cf. [27]). A situation is best thought of as a fragment of a world, or a partial description of a world. Situations are similar to possible worlds, but differ in that whereas a world assigns to every primitive proposition either *True* or *False*, a situation may assign it *True, False, Neither* or *Both* (in which case, a situation is said to be *incoherent*).

Levesque proceeds by assigning an agent a set of situations, which, somehow, the agent considers possible. Intuitively, an agent could be said to explicitly believe ψ if ψ were assigned the value *True* in each situation of the considered set.

More formally, a model of Levesque's logic is determined by three elements: a set $\mathcal{B}$ of situations (with $\mathcal{B} \subseteq \mathcal{S}$, the set of all situations), which turns out to identify the agent's explicit beliefs, and two valuation functions, $\mathcal{T}$ and $\mathcal{F}$, which take a primitive proposition 'p' and return the sets of situations $\mathcal{T}(p)$, $\mathcal{F}(p)$ (subsets of $\mathcal{S}$) in which 'p' is, respectively, assigned a *True* and *False* value.

Given a model and a specific situation 's', the truth of a proposition 'p' is said to be '*supported*', in that situation, iff $s \in \mathcal{T}(p)$. Similarly, its falsehood is supported iff $s \in \mathcal{F}(p)$. Finally, a formula ψ (built, as usual, with propositions and standard connectives '$\neg$' and '$\vee$') is (explicitly) believed by the agent (formally, $B\psi$) iff it is supported in *every* situation present in the set $\mathcal{B}$ of the model.

Now, consider the following aspect of the problem of logical omniscience: because of the necessitation rule, in possible worlds semantics, an agent is supposed to believe all the *tautologies* of propositional logic, as they are valid formulae.

Unfortunately, the set of the logical tautologies is infinite. How does Levesque's formalism overcome such problem?

The answer lies in the fact that, in possible world semantics, *every* world acts as a 'propositional valuation', that is, assigns the truth value *True* to all the tautologies, whereas a tautology is *not* necessarily assigned *True* in every situation in $\mathcal{B}$. In short, situations do not act as propositional valuations, so the problem of infinite (explicit) beliefs is avoided.

Finally, in order to complete his logic, Levesque introduces the implicit belief operator (L), such that a proposition ψ is implicitly believed ($L\psi$) by the agent in a situation s iff its truth is supported in *all* the situations of $\mathcal{S}$ which 1) 'agree' with 's' on the truth or falsity of propositions, and 2) act as classical propositional valuations. With this definition, L has the properties of a normal modal logic necessity operator $\Box$ with axioms KT5.

A number of objections have been raised to Levesque's model (cf. [124, p.135]): first, it does not allow nested beliefs; second, his notion of a situation might be considered even more obscure than the notion of a world in possible worlds; third, Fagin and Halpern [37] have shown that under certain circumstances, an agent modelled by Levesque's scheme must still be aware of propositional tautologies.

To overcome this negative result, Fagin and Halpern have themselves developed a logic of 'general awareness' ([37]) based on a similar idea to Levesque's but with a simpler semantics; however, this proposal has been criticised by Konolige in [83], where he concludes:

> "It does not seem that there is much to be gained by considering a logic of general awareness, at least as far as modelling resource limited reasoning from beliefs is concerned. It is no more powerful than [the deduction model], and can be re-expressed in those terms." [83, p.248]

Hence, it seems appropriate to examine Konolige's alternative, which falls in the category of *sentential* models of beliefs which use a modal language.

The Deduction Model of belief

The starting point of Konolige's analysis lies in the following observation: "The deduction model was developed in an effort to define accurate models of the beliefs of AI robot planning systems." [82, p.3]. Thus, the aim of his work was to develop a model of beliefs of artificial, computational agents, and not a model of human believers.

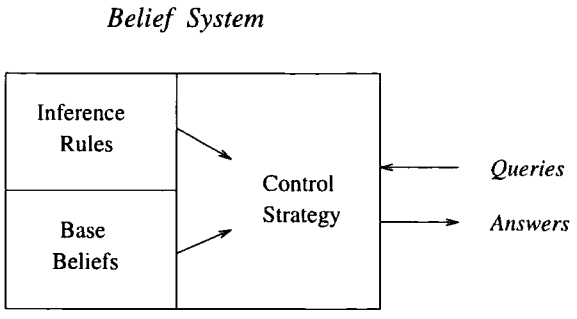


Figure 2.1: Konolige's schematisation of a typical AI belief system.

He argues that the two most important properties of such a model are:

- agents can draw conclusions from an initial set of beliefs, but
- they do not necessarily derive all the logically possible ones.

Possible worlds logics of belief capture the first property; they do not capture the second. Konolige proceeds by noticing that a typical belief system has three key components: a database of symbolically represented beliefs, a group of inference rules (*deduction* rules) and a control strategy (see Figure 2.1, taken from [82, p.19]).

> "When presented with a query, the belief system has a fixed, short amount of time within which it tries to answer the query, either by *matching* it directly to some fact in the base set, or by applying inference rules to derive it indirectly. The *belief set* 'bel(M)' is the set of sentences for which the belief system M answers 'yes'." [*ibid.*, p.19]

With respect to the problem of contradictory beliefs, Konolige describes two possible types of *inconsistency*: when both the formulae ϕ and $\neg\phi$ are in the belief set 'bel(M)', a belief system is simply said to be contradictory. However, when the *base* set is logically inconsistent, but the inference rules are not strong enough to derive a simple contradiction, the system will evaluate true both of the mutually inconsistent sentences ϕ and ψ, but will evaluate false both $\neg\phi$ and $\neg\psi$. Hence, the belief system is logically inconsistent, but not contradictory.

An important feature of Konolige's formalization is that it distinguishes between *internal* and *external* language of beliefs: while the 'base beliefs' contains a set of statements expressed in the internal language (generally a logical one), the external language is used by the 'observer' to describe the beliefs of the agent or to *query* the system.

It is important to notice that this distinction is only intuitively analogous to

that of 'object' language and 'metalanguage'; as Konolige points out,

> "...we could use a metalanguage/object language structure to express this distinction. The external language would have terms that refer to sentences of the object language, and a *predicate* $Bel(\ldots)$ [...]. However, [...] the metalanguage is notationally complex. For example, there must be metalanguage functions that construct all of the different boolean and quantificational expressions allowed by the internal language.
> Instead, we follow the simpler course of defining a belief *operator* whose argument is a sentence or sentence schema of the internal language." [*ibid.*, p.28]

Moreover, Konolige also argues that an agent uses sentences of his internal language to refer to the beliefs of other agents: he introduces the idea of a (modal) belief operator in the *internal* language, "an operator whose arguments are an agent S and a sentence ϕ, and whose intended meaning is that S believes ϕ" [Op.cit., p.27].

With respect to the application of inference rules, Konolige proposes to simplify the modelling process of the control strategies by assuming that an agent possesses a *finite* set of rules, which are applied exhaustively to the agent's base beliefs. An agent then believes ϕ iff it is possible to derive ϕ from the agent's base beliefs using the deduction rules, that is, if ϕ belongs to the deductive closure of the set of beliefs.

Formally, the deductive closure can be defined as a function '*close*':

$$close(\langle \Delta, \rho \rangle) =_{def} \{\phi \mid \Delta \vdash_{\rho} \phi\}$$

where $\langle \Delta, \rho \rangle$ constitute a *deductive structure*, composed of a set Δ of internal language formulae of belief and of a set ρ of deduction rules. The notation $\Delta \vdash_{\rho} \phi$ means that ϕ can be proved from Δ using only rules in ρ.

Konolige defines two modal languages for describing beliefs: L^B, which does not allow quantifying a variable which appears inside a modal context from outside that context, and L^{B_q}, which does. The language L^B is initially defined as external, and is based on an internal language L_0, which is a first-order language. However, as Konolige points out, if the internal language needs to allow the representation of other agents' beliefs, "[t]he natural candidate for the internal language L is then L^B itself." [*ibid.*, p.32].

It should be noticed that the model defined includes the possibility for an agent

to be *undecided* about a specific formula ϕ; such case occurs when neither ϕ nor $\neg\phi$ belongs to the current deductive closure.

Finally, Konolige also shows that the deduction model can be treated as a generalization of possible worlds semantics, and that any of the 'standard' possible worlds systems can be modelled using the deduction model (cf. [*ibid.*, Chapter 6]). Furthermore, he develops a variety of proof methods for his logics, including resolution and tableau systems [57].

To conclude, while Konolige's model is arguably too simplistic to capture the intricacies of human belief, it represents a very interesting model of the beliefs of AI systems, and is therefore relevant for the purposes of this thesis.

Belief revision: RMS and AGM

An important aspect of the theories and implementation of belief systems consists of the problem of *belief revision* and truth maintenance.

A system of beliefs is not normally seen as a 'static' set of beliefs, but rather as a dynamic system which must be ready to accept new information in input. The addition of a new belief to the system might cause the rise of a contradiction, or the change of attitude towards other beliefs which are supported (or challenged) by the new information. On the other hand, the removal of an old belief which has become obsolete can generate the consequent lack of support for other beliefs, which, in turn, might become uncertain, leading to a 'chain' propagation of the process of revision.

Researchers in this field seem to have reached a consensus on the fact that, after a change has been effected on the beliefs set, the revision process should be aimed at restructuring it in order to maintain intact certain required *properties* of the system. The two main properties with which they have been concerned are *consistency* (or *coherence*) and *groundedness*.

The notion of 'consistency', in its stronger connotation, requires that a system contains no logically contradictory beliefs. However, as already seen in Konolige, there are also weaker notions of consistency, which require only no 'obvious' contradiction, but admit possible 'derivable' inconsistencies which are not discovered by the agent because of the limited resources.

The concept of 'groundedness' is normally intended to refer to the structure of justifications, or *reasons*, which constitute the basis of the beliefs: again, in its stronger form, this means that *every* belief present in the system should be

supported by appropriate evidence.

The work of Harman [66], discussed in Section 2.2.1, presents two extreme possible approaches to belief modelling and revision, each of which puts more importance on one of the two above properties; they have been identified, respectively, as the *foundations* approach and the *coherence* approach (see pag. 22).

The notion of *Truth Maintenance System* (TMS) or, as subsequently amended, *Reason Maintenance System* (RMS), introduced by Doyle in [28], has been viewed by many (e.g. [66], [55]) as a classical example of foundations approach. According to Doyle ([29]),

> "...the foundations approach divides beliefs into two classes: those justified by other beliefs, and those not justified by other beliefs. The former constitute derived beliefs, while one may view the latter as "self-justifying" (or non-inferential) beliefs. The model provides 'explanations' of beliefs by requiring that each derived belief be supportable by means of some non-circular arguments from basic beliefs. This non-circularity also allows one to use arguments to determine changes to the overall set of beliefs; one should retain a belief, even after removing one of its justifications, as long as independent justifications remain [...]."[*ibid.*, p.34]

On the other hand, the coherence approach to belief revision maintains that an agent holds some belief just as long as it agrees with the agent's other beliefs, independent of how they may have been inferred or adopted. While one belief may be related to more beliefs than another, no belief is more fundamental than another. The so-called AGM theory of Alchourrón, Gärdenfors and Makinson [3] endorses the coherence approach, and is based on a 'conservative' approach to belief revision.

Let us analyse briefly these two different models just introduced.

Doyle's RMS can be said to be a *semantic-network-based* model of epistemic states. A semantic network typically consists of a set of *nodes* representing concepts (i.e. propositions, object of belief) and of a set of *links* between the nodes which represent 'relations' between concepts (cf. [134], [88]).

In RMS, there are two basic types of entities: *nodes*, representing propositional beliefs, and *justifications*, representing reasons for beliefs. Justifications may be other beliefs from which the current belief is derived. Since each specific node can

have associated various justifications, these can be thought of as (labelled) links to (or from) other nodes.

A node may be *in* or *out*, which correspond to accepting as *True* or *False* the belief it represents. If a certain belief is 'out' in the system, the corresponding negation does not necessarily have to be 'in'. On the other hand, if both a belief and its contrary are 'in', then the system will start a *revision* of the set of nodes and their justifications in order to re-establish consistency.

The justifications of a node are represented as a pair of lists: an 'inlist' and an 'outlist'. A node is in if and only if it has *some* justification, the inlist of which contains only nodes that are in and the outlist of which contains only nodes that are out. In other words, the inlist contains reasons *supporting* the belief, whereas the outlist contains the possible 'objections' which have been 'ruled out' (cf. the concept of knowledge in Ackermann, Section 2.2.1).

A particular kind of justification, called "nonmonotonic" justification, is introduced when the system does not have enough *positive evidence* to support a node, but has sufficient evidence to 'attack' its negation. For example, a belief in 'p' can be justified simply by the fact that the belief in $\neg p$ is out. This technique represents a way of modelling commonsense '*default*' expectations. It uses nonmonotonic reasoning in the sense that, given the previous example, a later *addition* of a (positive) reason for $\neg p$ will lead to a 'retraction' of the belief in 'p'.

Finally, a state (N, R) is a *legal* state of RMS iff N consists exactly of the *grounded* consequences of the reasons (justifications) R.

In contrast to RMS, the AGM approach models states of belief by sets of propositions. In some treatments (e.g. that of Gärdenfors [54]) states of beliefs are modeled by deductively closed but not necessarily consistent sets of propositions. In other treatments, however, states of belief are modelled by sets of propositions that need not be deductively closed; these sets are normally called *belief bases*. Many of the theoretical results about belief revision concern the former type of models.

The AGM theory considers three types of operations on belief states. For each belief state K and proposition 'p', we have:

Expansion: Expanding K with p, written $K+p$, means adding p to K and requiring that the result be a (possibly inconsistent) belief state;

Contraction: Contracting K with respect to p, written $K \dot{-} p$, means

removing p from K in such a way to result in a belief state;

Revision: Revising K with p, written $K \dot{+} p$, means adding p to K in such a way that the result is a *consistent* belief state.

Expansion is naturally defined in terms of the union of the set of beliefs and the new proposition. Contraction and revision, on the other hand, have no natural definitions, only the standard requirement that the change made be as small as possible so as to minimize unnecessary loss of knowledge and resources. This requirement does not define these operations since there are usually several ways to get rid of some belief. Nevertheless, Alchourrón, Gärdenfors and Makinson formulate and motivate sets of rationality postulates that these operations should satisfy [3].

Although Harman and Gärdenfors regard these two approaches to belief modelling and revision as antithetical, Doyle argues in [29, p.30] that they differ "... less than has been supposed, in that the fundamental concerns of the coherence approach for conservativism in belief revision apply in exaclty the same way in the foundations approach." In fact, as he clarifies later on, "the RMS update algorithm was designed to attempt to update states conservatively. That is, if one takes the current state (N, R) and modifies R to obtain R', RMS should choose a new legal state (N', R') with a set of nodes N' as close as possible to the current set N" [*ibid.*, p.35].

Moreover, because of the main objection raised to the foundations approach, namely, that it involves excessive computational expense for bounded resource agents, Doyle has more recently introduced the idea of *local* coherence and groundedness (cf. [30], [31]).

Although retaining the idea that a reason maintenance system keeps track of what information has been computed from what, he takes the purpose of the RMS to be "to maintain a description of the overall system's state of belief that is *as good as possible* given the reasoner's purposes and resources" [31].

To make the RMS adaptable to rational control of the effort expended on revisions, Doyle divides the belief system into parts, called *locales*, each of which may be revised or preserved separately. Each locale contains its own set of beliefs, and maintenance/revision instructions are defined in relation to the locales of the system. These instructions may indicate that changes should propagate within the locale containing the belief, or to its neighbours, or globally; or that all beliefs in the locale should be grounded with respect to the locale, with respect to its

neighbours, or globally.

Because of this partiality, the overall set of beliefs may exhibit inconsistencies by including conflicting beliefs from different locales; moreover, the RMS will be sometimes unable to track all the consequences of all beliefs, which will result to be not fully grounded. However, specialized locales corresponding to specific problems will be kept grounded in the axioms formulating these problems: "[t]he system of beliefs can thus be thought of as 'islands' of groundedness floating in a sea of ungrounded beliefs." [31].

This concept clearly fits into the idea of a cognitive state in which multiple belief systems exist in relative isolation from each other, present in Armstrong's (and Stalnaker's) work (see p. 17). Moreover, as in the views of Ackermann, Armstrong and Harman, at the basis of Doyle's work lies the idea that the amount of sufficient evidence which should be kept as a support for the beliefs of a bounded resource rational agent is highly *context-dependent* and essentially *subjective*.

2.3 Discourses and arguments

The definition of a theory of discourse is one of the main problems of the field of linguistics. The two main facets of this problem, discourse *analysis* and discourse *generation*, are strictly related to each other.

It would seem quite straightforward that different models of discourse generation lead to different approaches to the problem of discourse analysis and understanding, and, thus, one could argue, the problem consists simply of deciding which is the most appropriate model for discourse generation, and the understanding bit will follow as a 'corollary'.

Unfortunately, since we do not have, in general, direct access to the mental processes which govern the construction of a discourse, one of the approaches adopted consists exactly of starting from the results — that is, from the available examples of actual discourses — in order to identify some 'clues' on the mechanisms underlying their generation.

Therefore, it is possible to identify two different approaches for the development of theories of discourse: in a "top-down" approach, a certain model of discourse generation is assumed, according to which specific discourses can be produced. The validity of the theory can then be tested by comparing the discourses which it can generate with the examples found in the real world, and, also, by testing the results of the application of such theory to the problem of (real world) discourse analysis

and understanding.

On the other hand, the "bottom-up" approach utilizes the material available to 'induce' a general theory of discourse analysis and evaluation. This approach represents also a fundamental source of 'inspiration' for devising models of discourse generation and, therefore, constitutes a useful 'feed-back' for the top-down approach.

Intuitively, a *discourse* can be defined as a *linearised structure of semantically related statements* generated by a single source in order to achieve a specific *communicative goal*[3].

As specified in Section 1.2, the scope of this thesis has been restricted in two ways: first of all, it has been limited to the 'high-level' aspects of the problem of automatic discourse generation, concerned with the formation of the basic structure of the message; the lower-level issues related to the actual 'surface-features' of the discourse and its realisation in NL will not be treated here. Secondly, the class of discourses considered has been restricted to that of *persuasive* discourses, that is, discourses which are generated with the goal of convincing the hearer about the validity or the fallacy of a specific proposition.

As a matter of facts, a consequence of the latter assumption is that other classes of linguistic structures can be included under the broad category of persuasive discourses. For example, with respect to the definition of *argument*, O'Keefe [104] distinguishes two meanings of this term: one refers to the structure created to support a case, whereas the other one to the activity of disputation. The former meaning can be considered intuitively equivalent to that of persuasive discourse, as defined above.

Moreover, consider the following quotation, taken from Reed and Long [120]:

> "One of the most prolific forms of persuasive argumentation, however, is the monologue [...]. Monologue [...] is a 'speaker-oriented' activity, relying solely on the speaker's knowledge, both of the domain of the argument, and also of the (presumed) *hearer* knowledge (of both the domain of the argument and the speaker, etc.). [...] [A] persuasive monologue has an aim — to alter the beliefs of an audience (most usually to convince [...])."[Op.cit.]

In this context, the form of persuasive argumentation considered — monologue —

[3]This definition is based on works which will be surveyed in this section.

seems to be analogous to the concept of persuasive discourse. Finally, according to Freeman [47, p.5], "Arguments *are* discourses in which certain statements are put forward to support others."

Thus, taking into account the above mentioned restrictions which apply to the scope of this thesis, it seems appropriate to include in this survey the description of some of the most representative theories concerning the structure of *discourses* and *arguments* which are relevant to the area of automated persuasive discourse generation; when recognizable, such models will be classified under the two broad categories of top-down and bottom-up approaches.

2.3.1 Discourse structure

According to Grosz *et al.*, the analysis of a variety of types of discourse (including arguments and explanations) has established that discourses divide into discourse segments, and that these segments may bear different kinds of relations to one another [62, p.439].

The question of how these segments can be identified is still the object of controversy in the field of computational linguistics; however, two major traditional approaches can be identified: the *formalist* and the *functionalist* perspectives (cf. [74]).

According to typical formalist analysis, discourse exhibits internal structure, where structural segments encapsulates semantic units that are closely related. Typically, the theories are used to explain pronominalisation and quantifier scoping effects. The theories tend to concentrate on the development of formalisms for discourse segments and the discourse structure itself, which usually is a tree with clause-level utterances or groups of them as the nodes. The theories tend to be weak on the actual contents of the structure, such as the precise interrelationships between segments and the communicative purposes of the discourse.

From the functionalist point of view, discourse exhibits internal structure, where the segments are defined by the communicative goals. The theories tend to concentrate on the intentions of the speaker and on the ways these goals are reflected in the discourse structure, often as interrelationships between segments (cf. [90]). The theories are strong on the particular intersegment relations, but tend to be weak on the *global* form of the discourse structure.

Some of the more influential formalist works are represented by Kamp's Discourse Representation Theory (DRT) [79], and by the works of Polanyi [111] and Cohen [23]. On the other hand, two representative examples of fuctionalist ap-

proaches can be found in the works of Hobbs [70] and Mann and Thompson [91], the latter being by far the most relevant in this context for its use in the field of discourse planning. Finally, a combination of the formalist and functionalist ideas is embodied in the theory of discourse developed by Grosz and Sidner [63], who describe an analysis of discourse based on three separate structures: the formalist segmentation of the utterances, the functionalist structure of the speaker's intentions, and the attentional state. Let us examine briefly the three most influential works, namely, those of Kamp [79], Mann and Thompson [91] and Grosz and Sidner [63].

Discourse Representation Theory (DRT)

Kamp's DRT [79] consists of a (bottom-up) two-step approach to a semantics for natural language. It provides an algorithm for mapping a fragment of English onto a system of representations and then defines a mapping from these representations into a first order model to determine truth conditions [9].

Central to DRT is the notion of Discourse Representations (DR), which are described by Kamp as mental representations which a person forms as a response to the input (s)he receives. Sometimes a representation will involve structured families of DRs (often representations of conditional sentences), and then this structured grouping will be called a Discourse Representation Structure (DRS).

Kamp claims that each sentence or discourse induces the construction of such a DRS, and only if the sentence or discourse is fairly simple will the DRS consist of only one DR. He adds that among the DRs within a DRS there is always one which represents the discourse as a whole.

An example of construction of DR is illustrated by the two sentences "Pedro owns Chiquita. He beats her." [79, p.284]. Symbols are assigned to the two elements contained in the first sentence, hence Pedro and Chiquita become 'u' and 'v', respectively. The DR produced is "u owns v". As a consequence of the second sentence, the DR produced is "u beats v".

The mapping from sentences — or rather, from their syntactic parses — into DRSs is performed following a *DRS construction algorithm*, which operates on the parse tree of a sentence in roughly a top-down manner, and is typically applied to such a parse tree within the content of an already established DRS representing the information provided by antecedent discourse, or, more generally, by the context of use. DRSs thus determine the truth conditions of a sentence 'S' when it is interpreted in isolation from its surrounding context, but they also reveal how the

context of S's use combines with the information contained in S (cf. [67]).

The DRS construction algorithm is intended to represent in a highly idealized way the processing of a verbal input by a recipient; the outputs of the DRS algorithm, the DRSs, are intended to represent the *information* that a hearer recovers from a verbal input.

Kamp's theory yields a treatment of indefinite and definite noun phrases that has made an important contribution to understand the anaphoric and "pseudo referential" behaviour of indefinites (cf. [9]). Even though this work has been considered relevant as representing one of the first attempts to take the discourse structure into account, the theory is extremely limited, as it does not examine motivations, communicative intentions and goals, which are considered to be at the basis of the production of the rich complexities of discourse phenomena.

Rhetorical Structure Theory (RST)

Mann and Thompson's RST [91] constitutes an analytical tool for describing the structure of text. It is based on a study involving some hundreds of paragraphs (ranging over advertisements, scientific articles, letters, newspapers texts, and others) and proposes a set of approximately 25 relations to represent the relations that hold within normal English texts.

The theory holds that the relations are used recursively, relating ever smaller segments of adjacent text, down to the single clause level; it assumes that a paragraph is coherent only if all of its parts can be included under one overarching relation. Most relations have a characteristic English cue word or phrase which informs the hearer how to relate the adjacent clauses; larger blocks of clauses are then related similarly, so that, eventually, the role played by each clause can be determined with respect to the whole.

Most relations contain two parts, a *Nucleus*, which identifies the major, central contents of the segment, and a *Satellite*, specifying more subordinate, qualifying material. Moreover, each relation has an intended *effect*, or function, which describes the expected consequences of that span of text on the hearer's attitudes and beliefs.

By way of example, the relation 'Evidence' (taken from [*ibid.*, p.251]) [4] is reported below:

[4] In the relation, 'N' and 'S' stand for Nucleus and Satellite, whereas 'R' and 'W' are the equivalent of hearer and speaker (respectively, Reader and Writer).

Relation Name:	EVIDENCE
Constraints on N:	R might not believe N to a degree satisfactory to W [...]
Constraints on S:	R believes S or will find it credible
Constraints on the N+S combination:	R's comprehending S increases R's belief of N
The effect:	R's belief of N is increased
Locus of the effect:	N

Although the interpretation of RST relations as *operators* for discourse planning (cf. [71], [74]) has produced results of significant popularity (work in this context will be described in more details in Section 2.4), one of the principal criticisms of this theory lies, once again, in the lack of specification of the intentional structure of the discourse (cf. [99]).

This problem can be seen as a consequence of the fact that RST was developed as a *bottom-up* approach to the study of the structure of the discourse, whereas the decomposition of communicative goals and intentions seems to be more a *top-down* concept of discourse structure generation for theories which assume discourse construction to be a goal-driven process. An important example of such approach is given by the work of Grosz and Sidner, which we now move on to consider.

The work of Grosz and Sidner

In the theory presented by Grosz and Sidner in [63], discourse structure is composed of three separate but interrelated components: the structure of the sequence of utterances (called the *linguistic* structure), a structure of communicative purposes (called the *intentional* structure), and the state of focus of attention (called the *attentional* state).

The linguistic structure consists of "segments of the discourse into which the utterances naturally aggregate." The intentional structure captures the purposes which lie behind the segments as well as the relationships among them. The attentional state is an abstraction of the focus of attention of the speaker and of the audience as the discourse unfolds. It is realized through a dynamic stack — derived from the linguistic structure — which records, for each segment, a list of objects, properties and relations that are *salient* at that point of the discourse.

The linguistic structure consists of the discourse segments and an embedding relationship that can hold between them, and which reflects the relationship among elements of the intentional structure. When a segment boundary is crossed and a speaker moves from one segment to another, cue phrases (such as "On the other

hand", "In contrast", etc.) can be argued as indicating a switch in the speaker *attention* or *intention*.

According to the theory, the discourse has an overall purpose (the *Discourse Purpose*, or DP), constituting the specific intention that underlies engaging in the particular discourse. Moreover, each segment is associated with a specific *Discourse Segment Purpose* (DSP), which specifies how this segment contributes to achieving the overall DP.

Three examples of possible intentions which could serve as DP or DSPs are quoted below (taken from [*ibid.*, p.179]):

1. "Intend that some agent intend to perform some physical task. Example: *Intend that Ruth intend to fix the flat tire.*
2. Intend that some agent believe some fact. Example: *Intend that Ruth believe the campfire has started.*
3. Intend that some agent believe that one fact supports another. Example: *Intend that Ruth believe the smell of smoke provides evidence that the campfire is started.*"

Notice that while the first two intentions concern an agent's (supposedly, the hearer's) belief towards a specific fact (case 2) and towards an agent's own attitude about performing an action (case 1), the last intention concerns the agent's belief in a *relation of support* between different beliefs.

Although the theory presented by Grosz and Sidner can be seen as a milestone in the study of the discourse structure, it is often argued that the main thrust of their ideas on focus and topic rely too heavily upon the syntactic features, and that their interest in the linguistic structure overshadows concern with intention (cf. [77, p.43]).

More recently, work has started on a principled unification of the tripartite distinction proposed by Grosz and Sidner and the Rhetorical Structure Theory (cf. [102]), whereby the notion of 'dominance' introduced in [63] is equated with that of nuclearity of the RST. Along this line is also the account given by Hovy, which is discussed below.

Hovy's computational definition

Surveying the text planning systems of several researchers for a variety of domains and taking into account various theoretical works, including that of Grosz and

Sidner [63], Mann and Thompson [91], Polanyi [111], and the work on intention recognition of Allen and Perrault [5] and others, Hovy presented in [74] a general formulation of plan-based English discourse; such formulation is summarized below, and concludes this survey on the works concerning the theory of discourse structure.

1. *Discourse.* A discourse is a structured collection of clauses. By their semantic relatedness, clauses are grouped into segments; the *discourse structure* is expressed by the nesting of segments within each other according to specific relationships. A discourse can thus be represented as a tree structure: at the top, the discourse is governed by a single root node; at the leaves, the basic segments are single grammatical clauses.

2. *Purpose.* Each discourse segment has an associated Discourse Segment Purpose, which is represented at each node of the tree (including the root). Each DSP consists of one or more communicative goals the speaker has with respect to the hearer's mental state. In a successful discourse, the contents of each segment achieve its DSP.

3. *Segment.* The content of a discourse segment is either:

   - an ordered list of discourse segments, together with one or more *intersegment discourse relations* that hold between them; or
   - a single discourse segment; or
   - the semantic material to be communicated, usually expressible as a single clause in English. This material often takes the form of a set of knowledge base assertions.

It should be noticed that this definition integrates the notions of intentional and linguistic structure introduced by Grosz and Sidner, and accounts for the representation of the RST intersegment relations proposed by Mann and Thompson.

2.3.2 Argument structure

As for discourse before, works on argument structure can be roughly divided into two main approaches, namely, bottom-up and top-down. In this context, the two approaches have led to the generation of *informal* structural models and analysis techniques (in the former case), and to the production of various *formal* models (in the latter case) aimed at extending standard logic representations to deal with the characteristics of naturally occurring arguments.

Both classes' most relevant and representative examples are surveyed in the two following subsections.

Informal approaches

The most common form into which arguments are analysed is an intuitive framework which shares the main principles with those contained in the foundations of the rhetorical analysis, which, ever since Aristotle, has considered arguments to consist of three basic elements: minor premiss; major premiss; *so* conclusion. (cf. [48]).

This *standard treatment* is well summarized by Freeman [47], who describes the four basic structures to be found in arguments as *divergent* (where one premiss con support several conclusions), *serial* (where a single premiss leads to a single conclusion, which may in turn constitute the single premiss to another conclusion, etc.), *convergent* (where two or more premisses contribute independently to a single conclusion) and *linked* (where two or more premisses together contribute to a single conclusion). These structures are graphically represented in Figure 2.2, as from [47, p.2].

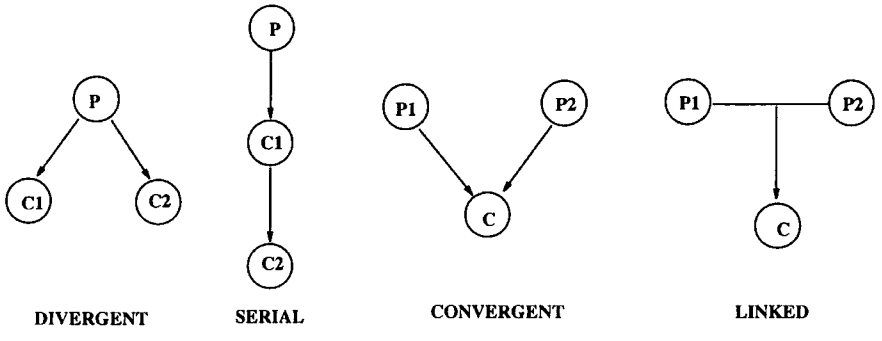


Figure 2.2: The four basic standard argument structures.

Complete arguments are composed of various combinations of these forms, where premisses to a super-argument can be the conclusion of a subargument.

The alternative representation of argument structure traditionally rival to the standard treatment consists of the theory proposed by Toulmin in *The Uses of Argument* [142]. Under Toulmin schema (see Figure 2.3) a claim, C, is supported by a datum D, qualified to a degree Q. The *support relationship* is, in turn, supported by a warrant, W, which is founded upon a backing B. Finally, an argument can include a caveat anticipating a rebuttal, R.

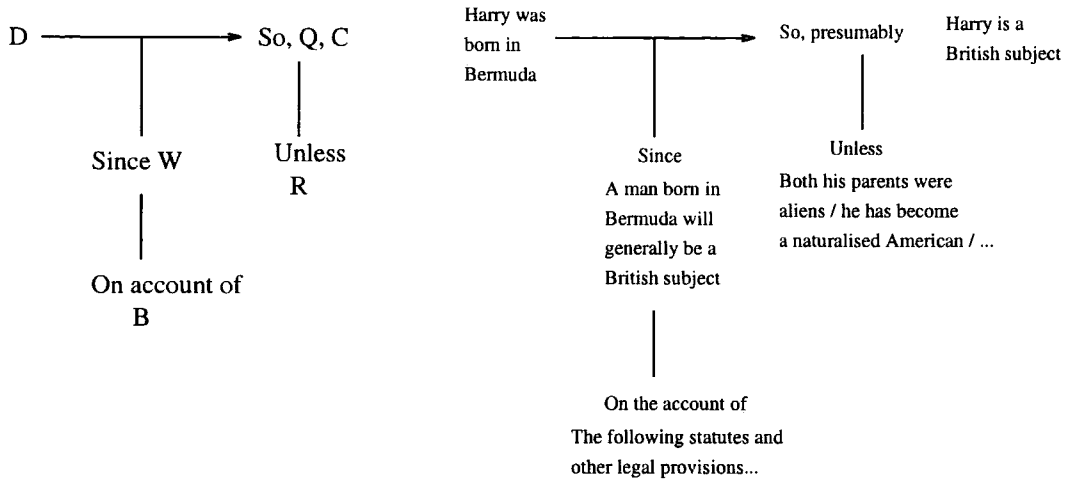


Figure 2.3: Toulmin schema (from '*The Uses of Argument*', p.104)

Toulmin introduces the notion of warrant by imagining our being challenged with the so-called '*question of relevance*', i.e. "Why is that reason relevant to the claim?", or "How did you get there?", posed after the assertion of the datum D and claim C [142, p.98] [47, p.38].

It should be noticed that the notion of 'warrant' represents, in an argument structure, the equivalent of a 'general belief' in the work of Armstrong (see Section 2.2.1). This concept is also mentioned by Grosz and Sidner in the example on possible Discourse Segment Purposes (see page 41), and by Ryle in [125]. As Freeman himself points out in [47, p.62], "Given an argument '*p, so q*' we may form its associated hypothetical, '*if p, then q*'. For the argument to be valid, the associated hypothetical must be true. [...] [W]e must admit in a candid formulation of the argument, the premise is '*p and (if p, then q)*', not just '*p*', as might be supposed." At this point, Freeman claims that the validity of the argument

Arg *p and (if p, then q), so q*

would require a further associated hypothetical, therefore producing a vicious infinite regress. However, he seems not to realize that the form of 'Arg' reflects that of the classical *Modus Ponens* rule, which is universally accepted without any further need for justification.

With respect to the concept of warrant, Freeman declares: "We should emphasize that for us, linked structure is exclusively connected with the question of relevance." [*ibid.*, p.94]. In other words, according to Freeman, the relevance question — at the origins of the presence of the warrant — gives rise simply to additional premisses in a linked structure. Hence, since no *structural* differentiation

exists between the elements linked, Toulmin's warrant and datum (and backing) are simply seen as different premisses of the same conclusion [*ibid.*, p.88].

In the standard representation, a convergent argument is one in which a conclusion is supported by disjunctive subarguments. However, according to Freeman, distinguishing between arguments which have a linked structure from those which have a convergent structure is particularly difficult a task. He considers, among others, the following example of real argument:

> La Petite Coloumb has the best chef in town. The live entertainment there is outstanding. The menu is also quite varied. Thus, we should go there for dinner. (from [*ibid.*, p.11])

He comments: "Each premise [...] gives us *some* reason for the conclusion, leading us to think the structure is convergent. But does any of the three, by itself, properly support, give us a good argument for, the conclusion? Would basing our decision to go to La Petite Coloumb for dinner on just one of these factors be hasty?" [*ibid.*, p.11].

In order to supply a representation for this type of arguments, Freeman introduces a further structure, derived from the convergent one by interconnecting the independent premisses with a horizontal line: "We might think of our separate, independent premises as weights which we place on one pan of a balance scale to support some conclusion." [*ibid.*, p.96].

Freeman is then led to believe that, since the premisses of a linked structure must be generated by a relevance question, and the two premisses A and B of the argument "Since A and since B, then $(A \wedge B)$" do not 'answer' that question, the new 'pan-convergence' structure should be used instead to represent *conjunctive* arguments as such.

Most of the research in argumentation theory is not, however, aimed at simply diagramming the structure of arguments, but, rather, at *evaluating* their validity. Adopting a taxonomy proposed by Blair [14], there are five basic approaches to argument evaluation.

Firstly, an argument is good if it is *logically* good, i.e. if its premisses are true and the inferences are deductively valid. This constitutes the original aim of logic as construed by Aristotle — logic having been developed to determine whether or not an argument is valid. However, this condition is generally too strong for

real arguments which often make use of non-deductive inference and of premisses which are *plausible* — or acceptable to all parties — rather than true. Secondly, an argument can be evaluated through the use of *schemes*, specific argument forms to which empirical data can be compared: each scheme is associated with a set of critical questions, by which validity can be determined. Thirdly, arguments might be assessed by reference to criteria of relevance, sufficiency and acceptability: premisses must be relevant to conclusion, must represent sufficient *support*, and must be *acceptable* to arguer and audience. Fourthly, argument validity can be equated with absence of fallacy. A list of the most common fallacies which, though illogical, are very common in natural language and are used (sometimes inadvertently) to strengthen an argument without actually supplying factual support, can be found in [118, p.159].

Finally, in a *subjective* and *context-dependent* view of argument evaluation, an argument can be defined valid if it is persuasive to a specific audience or hearer (cf. [108]). This evaluation technique must take into account issues of effective communication in real situations, and consider the various socio-psychological factors which affect natural argumentation (cf. [118, p.31] for a discussion on these aspects).

Wilson's '*The anatomy of Argument*' [147] represents a typical example of texts on argument evaluation. It contains most of the elements listed in Blair's taxonomy. For example, Wilson considers both the validity of *deductive* and the strength of *inductive* arguments, and the relevance, support and acceptability of the premisses. He develops a schema consisting of nine separate steps which should be followed to fully evaluate an argument. Finally, he also considers the conditions for the successful communication of an argument in terms of hearer's and speaker's obligations with respect to issues such as the attention, the communication of the structure and the problem of recall [147, p.347].

Other similar accounts on the representation of the structure of arguments and their assessment can be found, for example, in Fisher [40] and Fogelin [41].

Formal approaches

There is an increasing interest in using argumentation for systems based upon formal logic which need to reason about the real world and, in particular, which must be able to cope with uncertain and incomplete information. Reasoning about such domains can seldom employ strict deductive inference; rather, it becomes necessary to use some weaker notion of '*support*' which, often, is associated to the

idea of *defeasibility* or non-monotonicity.

An interesting example is represented by the 'Logic of Argumentation' (LA) theory, proposed by Krause *et al.* [84], which uses a labelled deductive system [49] to record sets of supports and determine acceptability. In LA, propositions are labelled with a representation of the arguments which support their validity. Arguments may then be *aggregated* to collect more information about the potential validity of the propositions of interest.

It is interesting to note that, as Krause *et al.* point out, in LA the "property of classical logic and intuitionistic in which support for the contradiction is propagated to all propositions is rejected. This was for several reasons. Primarily we wish to be free to choose how, or whether, contradiction should be resolved; the significance of a level of contradiction may differ in different problem solving contexts." [84, p.130].

Finally, the authors claim that the theory "provides a uniform framework which incorporates a number of numerical and symbolic techniques for assigning *subjective* confidences to propositions on the basis of their supporting arguments" [*ibid.*, p.113]. This approach, motivated by the need to reason under uncertainty (cf. [34]), has successfully been applied in a number of medical domains [43].

Another successful formal approach to dealing with uncertain and incomplete information is argument-based *defeasible reasoning* (see, for example, [113]). An interesting example of this approach is represented by the work of Dung [32], who presents a formalization of the idea of 'acceptability' of an argument.

The main component of Dung's formal theory is represented by an *argumentation framework*, consisting of a pair $\langle AR, attacks \rangle$, where AR is a set of arguments, and *attacks* is a binary relation on AR. Intuitively, an argument A attacks another argument B iff it supports the negation of B. Given a specific argumentation framework, an argument $A \in AR$ is *acceptable* with respect to a set S of arguments iff for each argument $B \in AR$ which attacks A, B is attacked by S. In other words, an argument is acceptable iff all the possible objection are ruled out by other arguments. It is interesting to notice that this notion clearly corresponds to the concept of *knowledge* as proposed by Ackermann in [1], which was discussed in Section 2.2.1.

Dung claims that the theory he develops demonstrates that many of the major approaches to non-monotonic reasoning and logic programming are different forms of argumentation. In his view, "... a defeasible statement can be believed only in

the absence of any evidence to the contrary, which is very much like the principle of argumentation." [*ibid.*, p.324].

In [114], Pollock agrees with Dung on the notion that an argument R is a 'defeater' for Q iff R is a reason for denying Q. However, he also points out that defeaters can be divided into *rebutting* and *undercutting*: whilst the former kind supports the negation of the claim, the latter kind attacks the *connection* between 'supporter' and 'supportee' (cf. [114][pp.379–380]). This is equivalent to attacking the 'associated hypothetical' of an argument, according to the description of Freeman [47] discussed above, or, in Toulmin's terms, it corresponds to attacking the thesis that the *warrant* supports.

Other representative examples of formal approaches to argumentation can be found in [144] and [116], the latter work being specifically oriented to the problem of reasoning in a 'legal' domain.

Recently, argumentation has been seen as a direct means of negotiation and persuasion between agents (e.g. [123]). Parsons and Jennings [105] extended the defeasible argumentation system of Krause *et al.* [84] such that one agent could propose an argument (for a specific course of action) to another, and the recipient would then evaluate the argument subjectively, that is, on the basis of its own beliefs, intentions and plans. The evaluation is performed by searching for rebutting and undercutting counter-arguments.

Nevertheless, as Reed and Long point out in [119], defeasible reasoning across rational, autonomous agents seems to miss the intuitions of how agents are functioning in reality. Following an account such as that of Dung [32] or Vreeswijk [144], a speaker agent S trying to convince a hearer agent H would communicate exactly those supporting arguments which together either defeat, or are undefeated by, all the rebutting and undercutting arguments that S presumes H to believe. However,

> "...in the real world, S may or may not know (or even be aware of the existence of) the [H's counterarguments] [...]. It is not the case, therefore, that S constructs her argument through anticipating H's possible counterarguments — rather, she is simply 'building a case' for her conclusion. Clearly, this process is going to involve consideration of what she thinks her hearer believes ([...]). But it does not require S to perform 'H-reasoning' to produce the arguments which she must ensure are defeated by her own." [119]

In conclusion, although the process of argument generation can be seen primarily

as showing evidence to support a claim, the possibility of including rebutting and undercutting defeaters in the model seems to be still necessary, as the presence of such arguments in the hearer's belief structure *could* have been actually detected by the speaker. In this situation, these beliefs would form a useful basis for the development of an alternative plan of persuasion in case of failure of the main one. This is the point of view adopted by Garagnani *et al.* in [51].

2.4 Planning

Planning constitutes one of the basic techniques adopted in the area of general problem solving. In a planning paradigm, states of the world are modelled as sets of propositions, and actions are modelled with *operator schemes*, representing general actions that can be performed under certain *conditions* and which lead to specific *changes* in the current world. Given a specific initial state of the world and a set of propositions representing the *goals* which have to be achieved, a correct *plan* solution consists of a sequence of actions (*steps*) which transform the initial state into one in which all the goals hold. Each action consists of applying an operator to a certain situation in which the specified conditions of that operator (also called *preconditions*) are verified; the changes produced by the action (i.e. its *effects*) lead to a new state of the world, in which another operator can be applied, and so forth.

According to the chronological analysis of the history of planning presented by Chapman in [20], at the basis of the development of this field lies the General Problem Solver (GPS) [103], which introduced the idea of *means-ends analysis*. The basic techniques that GPS adopted consists of adding an operator to a plan to achieve some of the current goals of the problem ('step addition') and taking the preconditions of the operator as new goals ('subgoaling') to be added to the remaining ones.

In 1971, through the implementation of STRIPS [38], Fikes and Nilsson introduced a general model of operator scheme in which the effects are specified by two lists of propositions (*add* and *delete* lists) representing the only changes that that operator will produce on the state to which it is applied.

Still according to Chapman, domain-independent conjunctive planning began in 1973 with Sussman's HACKER [137], which contained many of the ideas which would have been adopted by later 'nonlinear' planners. For example, the concept of 'promotion', which consists of constraining one operator to follow another in a

plan in order to avoid conflict ('clobbering') between the effects of the former and the preconditions of the latter.

The first truly nonlinear planner was Sacerdoti's NOAH [127]. The important idea which NOAH introduces is that a plan (at least while it is being constructed) does not have to specify fully the order of execution of its steps. In other words, a plan becomes only a *partial order* on steps. NONLIN [140] extends NOAH by adding backtracking to enable a better search for a correct plan solution. In MOLGEN [135], Stefik formalised the concept of 'constraints' (order and codesignation) imposed on a partial order plan; Wilkins' SIPE [146] was based on the same idea, but incorporated various new techniques.

This group of planning systems is followed by Chapman's TWEAK [20]. Chapman bases his work on the formulation of a criterion (the so-called *Modal Truth Criterion*) which establishes the conditions that guarantee a proposition to hold (necessarily or possibly) at a specific point of a partial plan. The algorithm which TWEAK implements consists of imposing the conditions specified by the criterion through a non-deterministic procedure.

Through this formalisation, Chapman succeeds in proving the completeness and correctness of his system. Nevertheless, as he points out in [20], this theoretical result is of little use in practice, as the "restrictions on action representation make TWEAK almost useless as a real world planner." [*ibid.*, p.350]. The restrictions Chapman is mentioning are essentially the following two: 1) the action representation does not allow the effects of operators to depend on the situation in which they are applied; and 2) all changes made by an action must be explicitly represented as effects of the operator.

The essential difficulty with extended action representation, as Chapman puts it, lies in the *frame problem* [76] [95]. The frame problem is traditionally stated as that of discovering what propositions are left unchanged by an action; thus, it can be viewed as corresponding to the problem of finding an efficiently implementable 'Modal Truth Criterion'. Unfortunately, in this respect, Chapman demonstrates the validity of the following negative result:

> "**Intractability Theorem** *The problem of determining whether a proposition is necessarily true in a nonlinear plan whose action representation is sufficiently strong to represent conditional actions, dependency of effects on input situations or derived side-effects is NP-hard.*" [*ibid.*, p.352]

Although this and other negative theoretical results "have caused some researchers to question the whole symbolic AI paradigm" [151, p.140], various attempts have

been subsequently made in order to improve the efficiency of planning systems, and, at the same time, to extend the expressiveness of the limited STRIPS representation.

For example, in the former category of attempts falls the idea of distinguishing, in the effects of an operator, between *primary* and *side* effects. As Fink and Yang point out in [39], the distinction can lead to significant computational savings by pruning the search space without compromising either soundness or completeness of the resulting planning algorithm. This idea has led, for example, to the development of systems which explicitly handle the links between one operator's postconditions and another's preconditions — 'causal' links — which could be threatened by the introduction of other operators in the partial plan. Partial order causal link (POCL) planners such as UCPOP [107] and SNLP [94], representing two examples of such approach, have been demonstrated to be sound and complete. UCPOP also addresses the restrictions on the expressiveness of the STRIPS language, adopting Pednault's ADL language [106] which permits the representation of conditional effects and universal quantification.

In order to cope with the complexity of real world domains, the proposal of using *abstraction* as a technique for improving the efficiency of planning (originally introduced by Sacerdoti in ABSTRIPS [126]) has also been reconsidered. At the basis of the notion of abstraction lies the idea of initially tackling the problem in a simplified representation of the world; once a solution has been found in such abstract level, the system can 'refine' it by moving to a more detailed level of domain description (cf. [44] for an account on hierarchical planning approaches).

An example of planning system which adopts a hierarchical approach and also exploits the advantages of a causal-link paradigm is represented by the AbNLP planner of Fox and Long [45] [46], in which operators are divided into 'abstract' and 'primitive'. An abstract operator consists of an external *shell* and of a *body*. The shell specifies the preconditions (*filters*[5]) and the effects (add and delete list) of the operator, whereas the body contains the list of goals which need to be achieved to realize the abstract action represented by the operator. In other words, the body specifies the goals which will have to be considered in the next level of plan-refinement. A primitive operator differs from an abstract one in that its body is empty.

Another approach to the improvement of planning performance is represented

[5]Notice that in contrast to the STRIPS operators preconditions, the mechanism of subgoaling is not allowed to be applied on filters.

by Prodigy [143], an architecture based on the idea of incorporating machine learning in planning and problem solving. Prodigy consists of a core general-purpose hierarchical planner and several *learning* modules that refine both the planning domain knowledge and the control knowledge to guide the search process effectively.

A different perspective on solving planning problems has been proposed more recently by Blum and Furst with 'Graphplan' [15]. In their system, rather than immediately embarking upon a search as in standard planning methods, the algorithm begins by explicitly constructing a structure called a '*Planning Graph*' which encodes the planning problem in such a way that many useful constraints inherent in the problem become explicitly available to reduce the amount of search needed.

A Planning Graph is a directed, leveled graph with two kinds of nodes (*propositions* and *actions* nodes) and three kinds of edges (preconditions, add and delete edges). The first level of a Planning Graph is a proposition level and consists of one node for each proposition in the initial state. The next level consists of all the possible actions (ground operators) that can be applied to the initial level; a further level of propositions follows — obtained as a result of the applications of the actions — which will contain all the propositions *possibly* true after one step (consisting of one or many parallel actions) has been executed. The procedure is then repeated on the newly created level of propositions, until either a level containing all the goals is reached and a plan can be found through a backward search through the graph, or a specific condition of termination is detected, indicating that the problem is unsolvable.

The Graphplan algorithm has been shown to be sound and complete, and to return always the shortest parallel-step plan solution, if a solution exists; the system's performance has been compared with that of different planners (including SNLP [94] and Prodigy [143]) and has produced competitive results in various domains (cf. [15]). However, one of the limits of Graphplan consists of the fact that it adopts an impoverished representation (namely, STRIPS formalism) for the operators description. To overcome this limitation, Koehler *et al.* [80] have produced a system (IPP) which extends the language of Graphplan to a subset of ADL [106], allowing conditional and universally quantified effects in operators and preserving soundness, completeness and the ability to terminate when the problem is unsolvable. IPP has been compared with Graphlan and other planners that support ADL; the results suggest that the system usually outperforms the others, and that the effected extension of the language does not lead to a computational overhead (cf. [80]).

Finally, an alternative approach to the problem of expressiveness versus manageability of a domain definition language has been presented by Gazen and Knoblock in [56], who describe a system to convert a UCPOP domain representation into a Graphplan representation. As the authors point out, a 'preprocessor' that translates domains from an expressive representation language into a simpler one for which more efficient planning algorithms exist is conceptually simple and not necessarily specific to one planner. As a matter of fact, this constitutes the approach which has been adopted in this thesis, although the solution proposed is partially different. The relevant similarities and diversities between this work and that of Gazen and Knoblock will be examined subsequently (Section 5.1.3), once the features of the system implemented will have been clearly explained and understood.

Having surveyed the work on planning from a general point of view, it is now appropriate to consider more specifically the *intersections* of this field with the other two previously examined, namely, that of belief systems and of persuasive discourses. Let us begin by considering the former subset.

2.4.1 Planning with beliefs: agents

One of the main areas where one would expect to find examples of integration between a belief system and a planner (apart from that of discourse and argumentation planning, which will be surveyed in the following subsection) consists of the field of research in *agents*. In fact, it seems natural to assume that some form of planning system will be a central component of any artificial agent. Moreover, it seems also natural to assume that agents will incorporate some form of belief system to keep a representation of the world in which they act.

However, in literature, many of the agent architectures whose main component is actually a planner are tailored for tasks which do not require the use of an actual language of beliefs, and such that the representation of the world consists essentially of their direct sensorial 'perceptions' (see [151] for a useful account on agent theories and architectures).

To this category of architectures belong, for example, Cohen's PHOENIX system [26] (which includes planner-based agents operating in the domain of simulated forest fire management), Wood's AUTODRIVE system [148] (where planning agents operate in a traffic simulation environment) and Etzioni's *Softbots*, that can plan and act in a UNIX environment [36].

Another class of agent architectures, adopting more sophisticated languages for the description of the internal cognitive state, is based on the work of Bratman *et al.* [17]. Their resource-bounded agent model relies upon the theoretical formalization of Rao and Georgeff [117], and exhibits four main symbolic structures: a 'plan library', actually containing the set of operator schemes, and explicit representations of Beliefs, Desires and Intentions (BDI architecture). In order to determine the plans to which the agent might decide to commit to achieve the current intentions, the architecture makes use of a specific means-ends analyser (planner) which is invoked to produce (structurally and temporally) *partial* plans and to further refine plans which had been left incomplete. The means-ends analyser bases the plan construction on the current set of beliefs, which is continuously affected by the *perception* of the external world and by the internal reasoning. This architecture has been evaluated in an experimental scenario known as the *Tileworld* [112].

A recent example of BDI agent architecture describing more explicitly the language for desires, goals, intentions, commitments and plans, can be found in the work of Brazier *et al.* [19], specified within a modelling framework for multi-agent systems (cf. [18]). In this model, the main emphasis is on static and dynamic relations between mental attitudes, considered at the basis of inter-agent cooperation. Beliefs, desires and intentions of an agent are modelled using a meta-language. Beliefs are divided into five different categories: internal beliefs, which an agent inherently has, with no further indication of their source; beliefs based on observations of the external world; beliefs based on communication with other agents (e.g. "if communicated-fact-by(X,A) and trustworthy(A) then belief(X)" [*ibid.*]); beliefs deduced from other beliefs; and beliefs based on *default* assumptions.

Although BDI architectures are interesting as they propose a possible model of integration between planning and belief modelling, they do not seem to take into consideration the cognitive state other agents, which, in the domain of discourse planning, plays a fundamental role. This is probably due to the fact that, in such architectures, the planning system is seen as a tool for generating plans for *practical action*, rather then plans aimed at producing specific changes in other agents' *mental* attitudes. Consequently, their belief languages do not normally involve nested or mutual belief expressions, nor allow the explicit representation of an hypothetical (hearer) agent model of beliefs.

Finally, somewhere in between the areas of planning agents adopting belief modelling and discourse planning lies the system developed for the 'TRAINS' project [6]. The project was aimed at building a plan reasoning system that

could communicate in natural language with humans in order to cooperatively solve generic tasks, and has been applied for the specific domain problem of train scheduling.

The functioning of the system is based on a dialogue exchange with a human agent (the 'manager'), on the basis of which the system tries to reach a definition of the problem. Once the problem has been identified, the system develops its own plan using a specific internal planner. Then, the information supplied by the manager are analysed and, through them, the system tries to *recognize* the plan which the manager is considering. Finally, the manager's plan is compared with the system's own plan and, in case of conflicts, the system tries to agree on a common plan during the dialogue.

The dialogue manager maintains a *mental state* in which it records the context of the conversation, the set of shared beliefs concerning the domain and the problem, and the plans considered so far, including the manager's ones. On the basis of the current mental state, communicative goals are generated, which will lead to state the endorsement or rejection of plans, or the proposal of new possibilities to the manager.

Although the system — through the recognition of the manager's plan — builds, to a certain extent, a 'manager model', there is no explicit representation of the hearer's beliefs, nor any adoption of a language of beliefs. Moreover, the process of discourse generation is simply reduced to transforming each communicative goal into a specific conversational act (such as request or assert) without actually involving any goal decomposition or discourse structure formation.

2.4.2 Discourse planning

Most of the past and current work on discourse and argumentation planning is conceptually based on the linguistic theories of *Speech Acts*, developed by Austin [13] and Searle [128].

Austin claimed that sentences in natural language are not simply propositions which can be evaluated true or false, but are in themselves 'acts'. More precisely, Austin distinguishes between *locutionary*, *illocutionary* and *perlocutionary* acts, all of which are present in any utterance. The locutionary act consists simply of uttering the words of the sentence; the illocutionary act can be seen as the act that the speaker performs (or, better, that (s)he *intends* to perform) in uttering the sentence (e.g. to inform, to ask, to apologize, etc.). The perlocutionary act consists of the *effects* that uttering the sentence has on the hearer's mental state. Quoting the

example given by Austin (cf.[*ibid.*, p.102]), a locutionary act is performed in saying "Shoot her!"; the illocutionary act (intended to be) performed by the utterance consists of '*Urging somebody to shoot her*', whereas the perlocutionary act would be '*Persuading somebody to shoot her*'.

Searle focuses upon the structure of illocutionary acts, and sets out precise *conditions* for their successful execution (cf. [128, pp.66–7]). These prerequisites make use of two types of constraint, concerning the *beliefs* and the *intentions* of the speaker. Searle also distinguishes between the hearer *understanding* a locution, and being *persuaded* about its validity, the latter being the perlocutionary effect described in Grice's analysis [60] and the former Searle's definition of illocutionary *effect.*

In order to clarify the distinction between the concepts of perlocutionary and illocutionary, effects and acts, communicating and understanding, it is appropriate to sketch briefly the process which occurs when a speaker tries to convey a structured message in NL to an hypothetical hearer.

The process can be intuitively divided into the following steps (cf. [131]):

(for the Speaker):

a) identify the internal concepts and the relations among them which will have to be communicated;

b) translate them into the 'equivalent' NL expressions, uttering them;

(for the Hearer):

c) *understand* the NL expressions (i.e. translate them back into an internal representation).

The completion of the communication process normally marks the beginning of a further phase of activity in the destination, representable as a step consisting of

d) inference

in which the hearer 'reacts' to the concepts communicated, reasoning on them, evaluating their validity and, possibly, creating new internal representations. Step *b*) corresponds to the locutionary *act*; step *c*) corresponds to the illocutionary *effect* (according to Searle), and step *d*) to the perlocutionary *effect.* The illocutionary act as defined by Austin seems rather to represent the speaker's intended *function* of the message, which could be seen as the speaker's perlocutionary *intention*, not

necessarily coinciding with the perlocutionary *effect*. Also, in Austin's view, step *c*) — understanding — was still part of the perlocutionary act.

The general consensus that linguists have reached on the importance of the role of perlocutionary intentions in the process of discourse construction has led recent work to concentrate more on the modelling of the cognitive state of the hearer. For example, in their analysis, Asher and Lascarides [10] "...assume I's cognitive state has embedded in it a model of A's cognitive state, which in turn has a representation of I's cognitive state", where A is the 'Author', and I the 'Interpreter'.

The point of conjunction between computational linguistic theories of discourse and planning can be taken to be the seminal paper of Cohen and Perrault [25]. Starting from the assumption that discourse generation is a goal-driven process, they developed a plan-based approach in which illocutionary and perlocutionary acts are represented as planning *operators* with specific preconditions (derived from Searle's theorisation) and specific effects, which included the expected changes of beliefs produced by the specific operator on the hearer. In order to specify such operators, Cohen and Perrault adopted a modal language of beliefs which allowed three levels of nested speaker-hearer attitudes (e.g. "Hearer believe Speaker Believe Speaker Want ACT", from [*ibid.*, p.190]). The plans produced contained direct requests, informs and questions; nevertheless, as pointed out at p.179 of [*ibid.*], they "...do not, however, discuss how those speech acts can be realized in words."

Cohen and Perrault also present a 'rudimentary' *Convince* operator, which relies on the idea that for an agent AGT1 to convince another agent AGT of the truth of a proposition p it is sufficient that AGT believes that p is believed by AGT1. Nevertheless, they immediately point out themselves the over-simplicity of this operator, commenting:

> "Though this might be a necessary prerequisite to getting someone to believe something, it is clearly not sufficient. [...] [B]efore AGT will be convinced, she needs to know the *justifications* for AGT1's belief, which may require that AGT believe (or be CONVINCE of) the justifications for believing those justifications, etc. Such a chain of reasons for believing might be terminated by mutual beliefs that people are expected to have, or by a belief AGT believes AGT1 already has. Ideally, a good model of CONVINCE would allow one to plan persuasive arguments." [*ibid.*, p.193]

Following the publication of this work, a number of systems adopted the planning

paradigm for discourse structure generation. A significant example is represented by Hovy's PAULINE, a discourse planning system which also integrates stylistic concerns [72]. Following Mann and Thompson's publication of the Rhetorical Structure Theory (see Section 2.3.1), Hovy adopted their RST relations for his paradigm, transforming them into planning operators [73] [74]. Hovy's planning algorithm makes use of 'growth points' for generating two spans of the discourse, based on the RST division between 'satellite' and 'nucleus' material. His formulation of the perlocutionary effects of the operators is expressed in terms of Cohen and Levesque [24] BEL and BMB (mutual belief) modal logic.

Another example of plan-based discourse and argument generation can be found in the work of Maybury [93], who proposes abstract plan operators which encode argument strategies such as '*convince-by-cause-and-evidence*'. The system developed by Maybury represents an agent's beliefs about their own knowledge and beliefs about other agent's knowledge at one level of recursion. However, the system does not infer the logical consequences of the current beliefs; as the author points out in [92], "... belief representation and modification was not a principal focus of this work".

All of the above models have been criticized (e.g. [99]) for their inability to record the *intentional* structure of the discourse which is being planned. To overcome this problem, Moore *et al.* [97] [100] [98] propose an alternative RST-based planning mechanism which distinguishes intentional from rhetorical structure. More precisely, their planner employs two kinds of goals: *communicative goals*, which aim to "affect the beliefs or goals of the hearer" [98, p.668], and *linguistic goals*, which characterise rhetorical relations. Fulfillment of communicative goals leads to the posting of linguistic goals, so the intentional and linguistic structures of discourse are built in tandem.

Although Moore *et al.* do recognize the importance of modelling the hearer's reactions to the speaker's utterances in the process of discourse planning ("... a generation system must represent and reason about the intended effect of individual parts of the text on the hearer [...]" [98]), the language adopted to express such effects is still based on single modalities such as BEL and KNOW-ABOUT, and does not address issues such as belief support or belief grounding (cf.[*ibid.*, p.671]).

A further step forward in the research on discourse planning is represented by the adoption of partial order causal links (POCL) planners instead of NOAH-based [127] architectures. This idea is implemented in LONGBOW [152], a discourse planning system based on the DPOCL (Decompositional POCL) planner,

developed by Young *et al.* [153] [154]. In this architecture, both causal links — between postconditions and preconditions — and decomposition (i.e. hierarchical) links are maintained. In a discourse context, such links are used to record the *intentional* structure of the discourse; they allow to distinguish between 'primary' and 'side' effects of an operator (i.e. of a 'discourse action'), and constitute fundamental information for the correct response of the system in case of plan failures (cf. [154]).

It is interesting to analyse briefly the '*Support act*' operator, reported below, which "... has the effect of increasing the belief in proposition *?prop* for the hearer" (from [153]):

Action

Header:	Support(?prop)
Preconditions:	not(Believe(?prop))
Effects:	Bel(?prop)
Bindings:	*none*

Decomposition

Header:	Support(?prop1)
Constraints:	causes(?prop2,?prop1)
Steps:	Start, Cause-to-believe-1(?prop2)
	Cause-to-believe-2(causes(?prop2,?prop1))
	Combine-belief, Final
Links:	⟨Combine-Belief, ?prop, ?prop, Final⟩
Bindings:	*none shown*
Orderings:	*none shown*

Notice the adoption of the relation 'causes( )', part of the informational structure, which expresses the fact that a proposition constitutes "a plausible cause of" another proposition. According to Young and Moore, "... one way to increase a hearer's belief in a proposition (i.e. to support a proposition) [...] *?prop1* is to find another proposition, *prop2*, such that causes(*?prop2, ?prop1*) [...]" [153]. The subplan contained in the decomposition is therefore a formalization of a strategy of 'persuasion' essentially based on a 'causal-support' version of the *Modus Ponens* rule, a concept already encountered in the informal work of Armstrong (see Section 2.2.1).

Although DPOCL is both sound and, for certain classes of plans, complete

(cf. [155]), its language for operator description seems to lack a clear separation between the model of the speaker and that of the hearer. Moreover, Fox [44] points out other shortcomings, which, according to Reed [118, p.27], are addressed in Fox and Long's AbNLP hierarchical planner. AbNLP has been adopted by Reed as the fundamental component of his 'Rhetorica' system for the automatic generation of arguments [121] [118]. In his system, Reed adopts a set of discourse planning operators, divided into a hierarchy of categories at the top of which lies the 'Argument Structure' level. In this class, Reed includes logical (Modus Ponens, Modus Tollens, Disjunctive Syllogism, etc.) and inductive operators, and, in addition, a group of fallacy-based operators. Rhetorica also allows the body of the operators to contain '*attentional*' subgoals, which deal with issues such as *focus* and the *ordering* of subarguments for persuasive effects.

Nevertheless, even in this architecture, the language of belief developed to specify the changes produced by the discourse operators on the hearer's mental state and beliefs is limited to very basic modal formulae, such as BEL(Ag, ~P) or the corresponding negation. Moreover, although Reed does presuppose in his architecture the presence of a model of beliefs containing a representation of the hearer's beliefs, the characteristics of such a system are not discussed.

In conclusion, all of the discourse planning systems considered lack a precise formalization of a language of beliefs and of its semantics; moreover, although generally endowing relations of *support* and belief justification, the models developed seem to neglect the issues of *consistency*, belief *grounding* and logical omniscience, normally associated with the introduction of a system of beliefs. Finally, the representation of the hearer model of beliefs is often not clearly distinguished from that of the speaker, and suffers from the same representational shortcomings. As Walker and Rambow point out in [145],

> "... effective text planners must explicitly model aspects of the Hearer's cognitive state, such as what the hearer is attending to and what *inferences* the hearer can draw [...]."

Thus, the definition of a formal *paradigm* for the specification of a speaker-hearer belief system and its use in the discourse plan construction is needed for the realization of complex, real-world persuasive discourse structures.

2.5 Criteria of evaluation

As mentioned in Section 1.3, the evaluation of the system presented here will be based on three fundamental parameters, namely, *expressiveness*, *efficiency* and *correctness*. Each of these parameters will be evaluated in two contexts: first of all, regarding the belief system as a tool designed for planning persuasive discourses; secondly, considering the belief system as a stand-alone piece of work.

By taking into account the various works surveyed in the previous sections of this chapter, it is possible to define more precisely how this evaluation can be performed in each of the two different contexts, which will be identified, in what follows, respectively as PD (*Planning Discourse*) and BS (*Belief System*):

Expressiveness:

PD: The model developed will have to be expressive enough to capture the relevant information needed during the process of construction of the discourse structure. These include:

1. the *intentional structure* of the discourse, i.e., the communicative goals which underly the generation of each discourse segment (Section 2.3.1);
2. the *functional relations* between different parts of a persuasive discourse described by the *standard analysis* of the argument structure (see Figure 2.2) and by the rival analysis of Toulmin (see Figure 2.3);
3. the *subjective*, that is, defeasible, or non-strictly logical, relations of *support* and *attack* between different sub-parts of an argument, the latter including *rebutting* and *undercutting* defeaters (see Section 2.3.2, Formal approaches);
4. the (hypothetical) point of view of the hearer on the relations mentioned in the previous point and on the objects of such relations (see Section 2.4.2).

BS: The expressiveness of the belief system should be evaluated by considering its ability to capture the fundamental elements found in *formal* and *informal* systems (see Sections 2.2.1 and 2.2.2). These include:

1. the ability to deduce new beliefs from existent ones according to assigned *rules of inference* and, yet, to avoid the problem of logical omniscience;
2. the ability to formalise and tolerate situations containing contradictory and *uncertain* information;

3. the ability to allow a *subjective* and context-dependent representation of a structure of belief *justification* and of *grounding* (coherence vs. foundations approaches).

Efficiency:

PD: The belief system developed will have to be used by a planner to produce discourse plans. The efficiency of this use will be evaluated in terms of:

1. *integrability* of the belief system, consisting of *i*) the complexity of the process of integration between the belief system and the planning system and *ii*) the range of planning systems into which the belief system will be able to be integrated;
2. efficiency of the final product, consisting of *i*) the impact that the process of integration has on the original efficiency of the planner and *ii*) the complexity of the maintenance of the integrated system;

BS: The efficiency of the belief system in itself can be estimated by the two following parameters:

1. *run-time* efficiency, calculated as the computational load required by the system to answer a query (i.e., to evaluate the doxastic attitude towards a specific proposition);
2. *maintenance* efficiency, i.e. complexity of the procedure of belief revision and reason maintenance.

Correctness:

PD: The correctness of the discourse planning system should be based on the formal proof of the correctness of the process of *integration* between the belief system and the planning system;

BS: the correctness of the belief system with respect to the given formal specifications will be based on the analysis of its actual implementation and on the results of a set of critical tests.

It should be noticed that the correctness of the discourse planning system as a whole follows directly from the last two conditions specified. In fact, assuming the

planning system adopted for the integration to be correct, if the belief system is in itself correct (as required by the second condition) and the process of integration is also correct, then the resulting final system must also conform the requirement of correctness.

Chapter 3

The Belief System Paradigm

This chapter introduces the theoretical paradigm according to which the belief system has been developed. The first section describes intuitively the model and its relationship with existing work. The second one formally defines the paradigm for belief systems, and presents some examples which show its suitability to deal with modal (doxastic) logic formulae. Finally, the last section contains some considerations on the complexity of the system.

3.1 Introduction

The paradigm for belief systems developed here belongs to the class of sentential, or interpreted symbolic structures models (see Section 2.2.2). In particular, it is based on the definition given by Ellis [33], according to whom a belief system is, in its simplest form, a set of assignments of the values *True*, *False* and X to the sentences of a language. Similarly, in this work a belief system is computationally defined as a calculable function (called 'Belief Prover', or BP function) which, given the current state — i.e. the set of beliefs currently held — assigns either the value *True* or *False* to each sentence of a language Lo (see Fig. 3.1). Notice that the use of the third value X in Ellis' model indicates the absence of any clear attitude concerning a sentence s; in this paradigm, the equivalent attitude of uncertainty towards s will be represented by imposing a situation in which neither s nor $\neg s$ is evaluated *True* by the BP function. This approach has also been adopted by Konolige in his 'deduction' model of belief (cf. [82]).

According to the previous definition, a belief system can be also seen as a 'black-box' whose behaviour is unequivocally described by a corresponding *formal theory*. In fact, given the set Lo of *wff*s, constituting the 'query' language of the system,

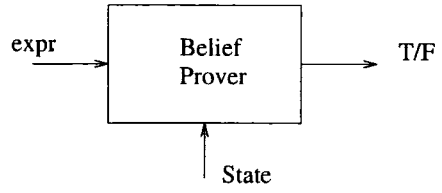


Figure 3.1: A belief system.

the current *state* of the belief system can be thought of as set of *axioms* of the theory. A certain query $s \in Lo$ will be evaluated *True* by the BP function *if and only if* s is a *theorem* of the theory, where the theorems of a formal theory consist of all the axioms plus all the *wff*'s which can be *derived* from the axioms through the use of a set of *inference rules.*

More formally, consider a finite set $R_1, R_2, \ldots, R_n$ of relations (inference rules) between *wff*'s such that for each relation R_i, there exists an integer $j > 0$ such that for any set P containing j *wff*'s and for any single *wff* 'C' it is possible to decide whether P is in the relation R_i with C. If this is the case, then C is a *direct consequence* of the set P of *premisses* through R_i. A *proof* is a sequence $A_1, \ldots, A_n$ of *wff*'s such that, for each i, A_i is either an axiom (i.e. $A_i \in X$) or a direct consequence obtained from the previous expressions $\{A_1, \ldots, A_{i-1}\}$ using one of the inference rules. A *theorem* of the theory is a *wff* A such that there exists a proof which contains A as last element of the sequence (i.e., A can be *derived* from the axioms).

This interpretation of belief system is also in accordance with the schematisation given by Konolige (see Figure 2.1), except for the fact that the 'Base Belief' (the state) is considered here as a *parameter* given in input to the system rather than as an integrating part of it.

Even though the behaviour of a belief system can be defined clearly through a formal theory, the actual *algorithm* for the theorem (or belief) proving, calculating the BP function of Figure 3.1, has not been identified yet. The definition of such an algorithm is clearly going to have an impact on the performances of the belief system. For example, suppose the algorithm consists simply of applying exhaustively the inference rules to the current state, as in Konolige's deduction model, and then checking whether the given query belongs to the resulting set of theorems: in this case, the inference rules cannot be strong enough to allow the generation of an *infinite* set of theorems.

In order to facilitate the definition of an efficient belief proving algorithm (or

BP algorithm, from now on) it is useful to investigate the nature and the origins of the *inference rules*, which characterise unequivocally the behaviour of a belief system.

Consider the set Lo of *wff*'s representing the 'query' language of a belief system. If Lo is expressive enough to allow an accurate and natural description of the doxastic attitudes, it is likely to be *redundant*, that is, to contain expressions which are *syntactically* different but *semantically* equivalent. More in general, a sophisticated language normally contains expressions which are related by *logical formulae*, determined by the semantics of the language. Such relations can be put in the form of 'inference rules' and used to *deduce* the truth value of expressions directly from the truth of other expressions of Lo. Therefore, by examining the logical relations 'characteristics' of a language Lo, it is possible to identify an '*inner*' language $Li \subset Lo$ such that Li contains all and only the expressions of Lo which are *semantically* unrelated.

The inner language Li is sufficient to describe *unequivocally* any possible state of the system: in fact, for any Lo, there is a (minimal) set of inference rules relating its *wff*'s such that any 'outer' state — defined in Lo — can be obtained from an equivalent 'inner' state (subset of Li) through the application of the rules.

Notice that the distinction between query language Lo and state-definition language Li is analogous to Konolige's distinction between *internal* and *external* language of beliefs, and is closely related to the concepts of *explicit* and *implicit* belief described by Harman and, more formally, by Levesque (see Sections 2.2.1 and 2.2.2).

Example 3.1 Consider the outer language

$$Lo = \{wife(s,t), husband(u,v), married(x,y), m(w), f(z)\}$$

where $m(\,)$ and $f(\,)$ mean, respectively, *male* and *female*, and x, y, w, z are meta-variables which can be instantiated to names. By giving to the expressions of Lo the common natural language interpretation, it is easy to identify the following semantic relations:

$$\begin{array}{lrcl} r_1) & married(x,y) & \rightarrow & married(y,x) \\ r_2) & married(x,y) \wedge f(x) & \rightarrow & wife(x,y) \\ r_3) & married(x,y) \wedge m(x) & \rightarrow & husband(x,y) \end{array}$$

This set of logical rules is *sufficient and necessary* to deduce the truth of the expressions '*wife()*' and '*husband()*' from that of the remaining expressions of Lo. Thus, the 'inner' language

$$Li = \{married(x, y), m(w), f(z)\} \subset Lo$$

can be adopted as state-definition language, as *any* (correct) outer state So defined in Lo can be 're-constructed' from an equivalent inner state $Si \subset Li$ through the applications of the inference rules. For example, the outer state

$$So = \{wife(Maria, Derek), husband(Derek, Maria), married(Derek, Maria), married(Maria, Derek), m(Derek), f(Maria)\}$$

can be directly re-constructed from (one of) its inner equivalent states

$$Si = \{married(Derek, Maria), m(Derek), f(Maria)\}$$

It is interesting to notice that in the previous example the inference rules, all in the form $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow c$, are such that $\forall i, p_i \in Li$. Moreover, the rules could be divided into two groups having consequences 1) $c \in Li$ or 2) $c \in (Lo \setminus Li)$. In other words, the inference rules could be splitted into 'inner' ones (r_1), used only to 'complete' the state with more inner expressions, and 'outer' ones (r_2 and r_3), whose consequences 'extend' the state to the language Lo. Because of this characteristic, their application to any given state $Si \subset Li$ to obtain the equivalent $So \subset Lo$ could be divided into two separate phases, consisting of *i*) exhaustive application of r_1 and *ii*) exhaustive application of r_2 and r_3 to the state resulting from the previous phase.

In summary, all of the previous considerations lead to the two following points:

1. *The language $\mathcal{L}_o$ adopted to query a (belief) system* contains *a simpler 'inner' language $\mathcal{L}_i \subseteq \mathcal{L}_o$ which can be used to define the current state of the system.*

2. *Given point 1, it follows that the definition of the BP function for the evaluation of an input expression must be such that the truth of each 'outer' query $w \in \mathcal{L}_o$ is calculated as a* function *of some expressions of $\mathcal{L}_i$.*

It follows immediately that the set of expressions obtained as the union of all the

domains of the functions constitutes the language $\mathcal{L}_i$. In what follows, the query language $\mathcal{L}_o$ of a belief system will also be indicated as '*outer*' language of that system, whereas $\mathcal{L}_i$ will constitute the '*inner*' one.

The concepts of 'query' (or outer) language, inner language and truth evaluation of each outer expression as a *function* of inner expressions are at the core of the formal definition of the belief system paradigm described in the following section, and will facilitate the definition of a sound linear algorithm for the integration of such systems into discourse planning systems.

3.2 The Belief System Paradigm

Let us assume an alphabet to consist of a set of letters or symbols.

Definition 3.1 A *Language* is a set of expressions (well-formed formulae) which are built — according to specific *syntactical* rules — using the symbols of a given alphabet.

Definition 3.2 Given a language L, a *State* is a set $S \subseteq L$.

Definition 3.3 Given a language L, a *Boolean Evaluation* of L is a decidable function $B : L \rightarrow \{True, False\}$.

A boolean evaluation simply maps every expression of a language into one of the two values *True* or *False*. The collection of expressions which are evaluated *True* by $B(\,)$ constitutes a state S representing the *set of beliefs* currently held by the system.

Definition 3.4 Given a boolean evaluation $B(\,)$ of L, the state *identified* by $B(\,)$ is defined as

$$S_B = \{w \in L \mid B(w) = True\} \tag{3.1}$$

In brief, the expression $B(w)$ corresponds to the question "Does w belong to the state?". Moreover, assigning a boolean evaluation $B(\,)$ of L is equivalent to specifying a state $S_B \subseteq L$. Analogously, given a state S, there exists only one possible boolean evaluation $B(\,)$ identified by S such that $S_B = S$.

The following definitions generalise and formalise the concept expressed in point 2 of the conclusive considerations of the previous section. In the formal model, the truth of any query will be evaluated as a '*boolean function*' of a finite *set* $\{a_1, a_2, \ldots, a_n\}$ of inner expressions:

Definition 3.5 Given a domain set $A = \{a_1, a_2, \ldots, a_n\} \subseteq L$ and a boolean evaluation $B(\,)$ of L, a *Boolean Function* of L — defined on A — is a *logical expression* $\mathcal{F}(B(a_1), \ldots, B(a_n))$ containing the operators '$\neg$' (logical negation), '$\vee$' and '$\wedge$', and the terms $B(a_1), \ldots, B(a_n)$.

For example, if $A = \{W_1, W_2\} \subset L$, the expression

$$\mathcal{F}_1(B(W_1), B(W_2)) = \neg\,(\,B(W_1) \vee B(W_2)\,)$$

is a boolean function (of L) defined on A. The logical value of $\mathcal{F}_1$ (either *True* or *False*) varies according to the boolean evaluation $B(\,)$ of L. In other words, when the current state contains W_1 or W_2, the expression $\mathcal{F}_1$ becomes *False*. For simplicity, a boolean function such as $\mathcal{F}_1$ will also be written as

$$\mathcal{F}_1(W_1, W_2) = \neg\,(\,W_1 \vee W_2\,)$$

assuming implicitly that every expression $w \in L$ will be substituted with *True* iff $w \in S_B$ (and, obviously, that w will be substituted with *False* iff $w \notin S_B$), being S_B the current state. This substitution will be formally represented by the expression

$$\mathcal{F}_1(W_1, W_2)\,|_S$$

indicating that the value of $\mathcal{F}_1$ is *calculated* in the state S. For example,

$$\begin{aligned} \mathcal{F}_1(W_1, W_2)\,|_{\emptyset} &= \mathit{True} \\ \mathcal{F}_1(W_1, W_2)\,|_{S_1} &= \mathit{False} \\ \mathcal{F}_1(W_1, W_2)\,|_{S_2} &= \mathit{False} \end{aligned}$$

where $S_1 = \{W_1\}$ and $S_2 = \{W_2, W_3\}$.

Let Lo be the outer language of a belief prover, and $Li \subseteq Lo$ be the inner language, used by the system to define the state. The following definition formalises

the idea of evaluating the truth of each query through the evaluation of the associated boolean function of inner expressions:

Definition 3.6 An *Interpretation* of *Lo on Li* is a function $\mathcal{I}$ associating each $w \in (Lo \setminus Li)$ with a *boolean function* $\mathcal{F}_w$ of Li, and each $w \in Li$ with the boolean function $B(w)$.

Notice that since the inner language Li is supposed to be used directly to specify the content of the state, the truth (or falsehood) of any inner expression will simply be determined by its presence in (or absence from) the current state. This is the reason why the interpretation function $\mathcal{I}(\,)$ associates every element of Li to the boolean function consisting of the element itself:

$$\forall\, w \in Li, \quad \mathcal{I}(w) = B(w) \tag{3.2}$$

where $B(\,)$ is the current boolean evaluation.

When containing contradictory information, the contents of the state can originate contradictory answers. In order to avoid this, we need to be able to *restrict* the range of possible configurations which a state can assume. The concept of 'boolean function' can be used to specify a condition on a specific set of expressions of a language:

Definition 3.7 A *Restriction* on a language L is a boolean function $\mathcal{U}$ defined on a finite subset $A \subseteq L$.

Definition 3.8 Given a restriction $\mathcal{U}$ on a language L, a state $S \subseteq L$ is *Coherent* with respect to $\mathcal{U}$ iff $\mathcal{U}\,|_S \;=$ *True*.

Example 3.2 Consider the finite language $L = \{a, b, c, d\}$. The restriction $\mathcal{U}$ defined on the set $A = \{\, a,\, b,\, c \,\}$ as

$$\mathcal{U}(a, b, c) = \neg\,(a \wedge b) \;\wedge\; \neg\,(a \wedge c) \;\wedge\; \neg\,(b \wedge c)$$

requires the three expressions of A to be *mutually exclusive*. Any state $S \subseteq L$ containing more than one expression of A will 'violate' the restriction $\mathcal{U}$. Hence, the only possible *coherent* states of L are $\emptyset$, $\{\, a \,\}$, $\{\, b \,\}$, $\{\, c \,\}$, $\{\, d \,\}$, $\{\, a, d \,\}$, $\{\, b, d \,\}$, $\{\, c, d \,\}$.

The choice of the term 'interpretation' in the previous definition is due to the analogy which can be found between this abstract structure and the characteristics of a natural language. The identification of a simple and *unequivocal* inner language $Li \subseteq Lo$ and the association of every element of $Lo \setminus Li$ with a *combination* of inner expressions is, in fact, analogous to the process of describing the unique *meaning* of every statement of a language (Lo) using a set of unequivocal terms (Li), i.e. of assigning them a specific semantics.

Definition 3.9 Given a language L, an *Inference Rule* r on L is an association between a boolean function $\mathcal{P}$ of L and an element $c \in L$, written as

$$r)\ \mathcal{P} \vdash c \tag{3.3}$$

Intuitively, if the left-hand side $\mathcal{P}$ (*premiss*) of the rule is evaluated *True*, then the right-hand side (*consequence*) of the rule should also be *True*, and therefore should be considered an element of the current set of beliefs. The inference rules are applied during the process of 'extension' of a state:

Definition 3.10 Given a state $S \subseteq L$ and a *set* of inference rules $R = \{r_1, r_2, \ldots\}$ on L, the *Closure* $\mathbf{C}_R(S)$ of the state S using the set of rules R is defined as

$$\mathbf{C}_R(S) = \{\omega \in L |\ S \vdash_R \omega\}$$

where $S \vdash_R \omega$ means that ω can be derived from S using only rules in R.

This definition is equivalent to that of *deductive closure* given by Konolige in [82]. Notice that, from the definition, it follows that

$$\forall S \subseteq L,\ \ S \subseteq \mathbf{C}_R(S) \tag{3.4}$$

It should be pointed out that the adoption of a mechanism of deductive closure does not necessarily lead the model to fall prey to the problem of logical omniscience; in fact, the set of inference rules adopted could contain only very limited and logically 'incomplete' rules, whose application can be carried out as a *finite* process of *practical completion* (extension) of the state, rather than one of *formal*

deduction. As a matter of fact, this constitutes the main reason which motivated the introduction of the closure mechanism in the model, as the description of the actual belief system implemented in Chapter 4 will clarify. In such a system, in fact, the actual 'inferential engine' will be represented by the mechanism of *interpretation*, defined previously, which allows the evaluation of the truth of any query without requiring explicit additions of beliefs to the state.

Example 3.3 Consider the language $Lo \leftarrow <Wo>$, defined as

$$\begin{aligned} <Wo> &::= \text{ABEL}(<Ei>) \mid \text{ABEL}(\text{NOT}(<Ei>)) \\ &\quad\ \text{ABEL}(<Wo>) \mid \text{NOT}(<Wo>) \\ <Ei> &::= \text{E}_1 \mid \text{E}_2 \mid \text{E}_3 \mid \ldots \mid \text{E}_n \end{aligned}$$

Let the set R of inference rules contain the following rules on Lo:

$$\begin{array}{lll} r_1)\ \neg B(\text{ABEL}(\text{E}_1)) & \vdash & \text{NOT}(\text{ABEL}(\text{E}_1)) \\ r_2)\ \neg B(\text{ABEL}(\text{E}_2)) & \vdash & \text{NOT}(\text{ABEL}(\text{E}_2)) \\ \vdots \quad\quad\quad\quad\quad\quad\quad & \vdots\ \vdots\ \vdots & \\ r_n)\ \neg B(\text{ABEL}(\text{E}_n)) & \vdash & \text{NOT}(\text{ABEL}(\text{E}_n)) \end{array}$$

In fact, R could also be represented as a single inference rule *schema*

$$r)\ \neg B(\text{ABEL}(e)) \vdash \text{NOT}(\text{ABEL}(e))$$

where the *meta-variable* 'e' can be substituted with any symbol generated by $<Ei>$.

Then, the closure $\mathbf{C}_R(S)$ of a state S will contain S itself plus all the "missing" propositions $\text{NOT}(\text{ABEL}(\text{E}_i))$. For example,

$$\begin{aligned} \mathbf{C}_R(\emptyset) &= \{\text{NOT}(\text{ABEL}(\text{E}_1)), \ldots, \text{NOT}(\text{ABEL}(\text{E}_n))\} \\ \mathbf{C}_R(\{\text{ABEL}(\text{E}_1)\}) &= \{\text{ABEL}(\text{E}_1)), \text{NOT}(\text{ABEL}(\text{E}_2)), \ldots, \text{NOT}(\text{ABEL}(\text{E}_n))\} \end{aligned}$$

Notice that the set R could even contain an (at most countable) *infinity* of rules. If these rules could be finitely represented, an algorithm which calculates $\mathbf{C}_R(S)$ for a given S would terminate and generate a *finite* output, if $\mathbf{C}_R(S)$ results to be a finite set.

The process of *completion* of the initial state through the application of a set of inference rules and the mechanism of *interpretation* constitute the two main components of the functioning of a *belief prover*, as the following definition shows:

Definition 3.11 Given a set R of inference rules on a language Li and an interpretation function $\mathcal{I}(\,)$ of Lo on Li, with $Li \subseteq Lo$, a *Belief Prover* (BP) for Lo on Li is an algorithm which calculates the function $BP : Lo \times 2^{Li} \rightarrow \{True, False\}$ defined as

$$BP(\omega, S) = \mathcal{I}(\omega) \mid_{S'}$$

where S' is the extended state $S' = \mathbf{C}_R(S)$.

Because of property 3.2, imposed by the definition of interpretation $\mathcal{I}(\,)$, the function $BP(\,)$ can be rewritten as

$$BP(\omega, S) = \begin{cases} \mathcal{I}(\omega) \mid_{S'} & \text{if } \omega \in (Lo \setminus Li) \\ True & \text{if } \omega \in Li \wedge \omega \in S' \\ False & \text{otherwise} \end{cases} \tag{3.5}$$

with $S' = \mathbf{C}_R(S)$. From the previous definition, it is possible to show the validity of the following property, concerning the function $BP : Lo \times 2^{Li} \rightarrow \{True, False\}$ calculated by a BP algorithm:

$$\forall S \subseteq Li,\ \forall \omega \in S, \qquad BP(\omega, S) = True \tag{3.6}$$

This follows directly from the definition of the $BP(\,)$ function: if ω belongs to the state S, then certainly ω is an inner expression; moreover, because of property 3.4, $S \subseteq \mathbf{C}_R(S)$, and, therefore, $\omega \in \mathbf{C}_R(S)$.

The outer language Lo, the inner language $Li \subseteq Lo$ and the specific BP algorithm adopted determine entirely the functioning of a *Belief System*:

Definition 3.12 A *Belief System* is a triple (Lo, Li, A), where Lo and Li are, respectively, the outer and the inner language, and A is a Belief-Prover (BP) algorithm for Lo on Li which calculates, for any given expression $\omega \in Lo$ and state $S \subseteq Li$, the boolean value $BP(\omega, S)$.

Example 3.4 It is interesting to show how the mechanism of outer language in-

terpretation can easily deal with *modal* logics expressions. Consider, for example, the rule of *necessitation* in the normal systems in modal logic (see [138]):

$$N) \quad \text{if } \models \phi \text{ then } \models \Box\phi \tag{3.7}$$

In other words, if ϕ is a *valid* theorem of the theory, then so is $\Box\phi$ (and, consequently, so is $\Box\Box\phi$, *ad infinitum*). In the belief system paradigm, the rule of necessitation can be realized simply by defining the outer language interpretation as follows:

$$\mathcal{I}(\Box\omega) = \begin{cases} B(\omega) & \text{if } \omega \in \mathcal{L}_i \\ \mathcal{I}(\omega) & \text{otherwise.} \end{cases}$$

where $\mathcal{L}_i$ is the inner language, and $B(\,)$ is the boolean evaluation of $\mathcal{L}_i$. In fact, this recursive definition constitutes the interpretation of an outer language $\mathcal{L}_o$ containing queries in the form $\Box\Box\ldots\Box\omega$, with an arbitrarily-high number of modal predicates '$\Box$' in front of any inner expression $\omega \in \mathcal{L}_i$.

3.3 Complexity and integration

In view of the process of integration between a belief system built according to this paradigm and a discourse planning system, it is appropriate to make some preliminary considerations regarding the computational complexity of the system.

First of all, if repeated queries $w', w'', \ldots$, are given to the system while the state S is left unchanged, the calculation of the closure set S' does not need to be repeated, and the real computational load of the system will be represented by the evaluation of $\mathcal{I}(\omega)$. This situation occurs when the number of modifications of the state S is much lower than the number of queries which the system is presented with. This is, in fact, the assumption which has been made at the beginning by describing the state of the belief system as a 'parameter' rather than as an input value (see Figure 3.1).

However, this assumption can be made only if the belief system is regarded as an independent, 'stand-alone' module. In fact, if the belief system is considered as a part of a discourse planning system, the frequency of state updates could be much higher, because of the use of *operators* which add and delete propositions to and from the state.

In order to reduce the work-load which repeated recalculations of the deductive

closure could involve, given a state S and its closure $S' = \mathbf{C}_R(S)$, it should be possible to develop a mechanism for the updating of S' which allows the identification, for a specific modification of the state S, of the limited *propagation* of changes in S' deriving from the consequences of the specific addition and/or deletion, without the need to effect the entire recalculation of S' from the new state.

In general, though, the realization of this mechanism would require a system to keep track of the relations of *dependency* between different propositions of the closure S', so that, for example, if a belief A can be derived from B using the inference rules, a modification of the state S involving B will produce a propagated effect involving the belief A.

Nevertheless, Chapter 5 will show that, under specific assumptions on the type of inference rules adopted, this 'localized' updating process can be realized without the use of explicit dependency links between the propositions of the completed state S', if the belief system is integrated within a planning system. This method has been actually adopted for the implementation of a discourse planning system which makes use of a belief system module built according to the theoretical paradigm defined in this chapter.

Chapter 4

A Speaker-Hearer Belief System

This chapter contains the description of a belief system for persuasive discourse planning, built according to the theoretical paradigm specified in Chapter 3. The model is intended to constitute the basic component of a 'Speaker' system which generates discourse plans for a specific audience (the 'Hearer').

In the first section of the chapter, an overview is given of the framework into which the belief system developed is intended to fit. The following sections are dedicated to the description and formalization of the belief system, whilst the final part contains a brief description of its implementation and illustrates some of the examples adopted for testing the system.

4.1 Overview

In this work, a discourse planning system has been considered to be a sub-part of a larger structure, namely, of an architecture which constitutes a speaker *agent*. The process of discourse structure construction has been assumed to make use of the *knowledge base* of the agent, as the contents of the discourse are not, in general, known *a priori*, and the system must be able to access any concept or information which might need to be included in the message.

The complete knowledge base of the agent has been assumed to be expressed as a collection of *events* which evolves through time. Intuitively, an event represents a fact, or an abstract concept, that can be held in our mind, such as "*The sky is blue*" or "*All men are mortals*". This conceptual unit represents, in semantic networks, the equivalent of a *proposition* in predicate logic[1].

[1]Semantic networks have been proven to be at least as expressive as First Order Logic (see

We can think of the actual process of discourse structure construction to occur, in one's mind, in an 'hypothetical' space, where only the relevant concepts are collected and organized, only the relevant relations between them (i.e. support, attack, etc.) are represented explicitly, and the (hypothetical) *effects* of the discourse on the hearer are planned. Since the process of discourse construction has been assumed to be *goal driven*, the speaker agent knows, from the beginning of the process, what communicative goals have to be achieved, and to which other concepts and beliefs such goals are related. Consequently, in the initial phase, the agent will 'scan' the entire knowledge base and extract the set of propositions and concepts which are *relevant* for the specific discourse communicative goals. This restricted and temporary version of the belief state will be used by the planner as *planning state.*

The state will have to contain only the information necessary for the discourse *structure* construction; hence, it will not need to represent explicitly all of the *natural language expressions* corresponding to the events/propositions extracted from the belief base, although it should be able to 'refer' to them. Therefore, the discourse planning process will treat the events contained in the state mainly as 'atomic' propositions (e.g. E_1, E_2, E_3, ...) whose semantic contents will not, in general, be accessible. This is because the declared aim of the system consists of producing discourse *structures*, without considering the issues related to the surface characteristics of the natural language realization. However, the features of the events and the relations between them which are useful for the representation of the structure of beliefs and for the construction of the discourse plan (e.g. support, grounding, rebuttal or attack) will be actually made explicit by the language of belief adopted, as will be shown in the subsection 'Features' of 4.2.1.

Once the communication phase has been completed, if the planned effects of the discourse on the hearer's beliefs correspond to those which were expected, the final (hypothetical) state reached at the end of the plan can be used to update the larger, underlying belief base of the speaker agent. If some parts of the plan did not achieve their intended aim, it will be necessary to identify the actual belief state of the hearer, update the larger belief base accordingly and start a new planning process, possibly re-using part of the previous plan.

The model described above can be considered as an underlying framework into

[134] and [130]).

which the system developed is expected to fit. The belief system described in the following sections of this chapter has been defined by taking such considerations into account.

4.2 Formalization

According to the definition given in Chapter 3, a belief system consists of a *triple* (Lo, Li, A), where Lo is the outer (query) language, Li is the inner (state content) language and A is an algorithm which calculates the function $BP : Lo \times 2^{Li} \rightarrow \{T, F\}$ defined as

$$BP(\omega, S) = \mathcal{I}(\omega) \mid_{S'}$$

where S' is the extended state $S' = \mathbf{C}_R(S)$ and $\mathcal{I}(\,)$ is an interpretation of Lo on Li.

Let us begin by defining the first element of the triple (Lo, Li, A), i.e. the outer language Lo.

4.2.1 The Outer language

Specifying the outer language corresponds to deciding which queries the system will have to be able to evaluate *True* or *False*. However, part of the expressions of Lo will be used to specify the content of the belief system, as the inner language consists of a *subset* of the outer one.

Moreover, it should be pointed out that this language has *not* been developed to fully represent the knowledge of an agent, but only to render explicit and make quickly available and updatable the *relevant* information required to construct a persuasive argument. As described in the previous section, the information contained in the state should be 'extracted' from a larger, complete, knowledge base (e.g. a semantic network) according to the specific *discourse intentions* (or 'communicative goals'), and then used for the discourse planning process.

Intuitively, the definition of the syntax for the outer language can be split into two separated parts: *attitudes* and *features*. Let us begin by considering the former type of expressions.

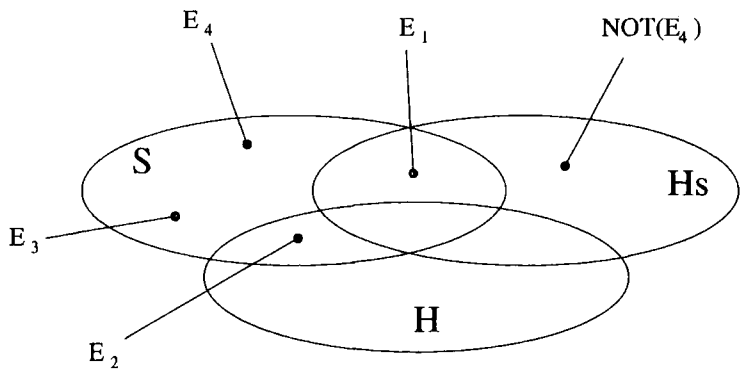


Figure 4.1: Graphical representation of speaker-hearer sets of beliefs.

Attitudes

The most obvious expression that any belief system should be able to deal with consists of the statement "the System BELieves proposition x", which, in our formalism, will be expressed as

SBEL(x)

where 'S' stands for System, or, equivalently, for 'Speaker'.

The belief model is intended to be used as a discourse planning tool by a system which generates discourse plans for an addressed 'Hearer', hence the necessity to represent not only the Speaker's beliefs, but also those events which are *assumed* to be believed by the Hearer.

Intuitively, the belief system can be thought of as an evolving set S of *events* believed by the speaker, as shown in Figure 4.1.

The set of events which the system assumes to be the audience's beliefs represents the *model of the hearer*, indicated by the set H_S.

For every event 'e' present in the set S of Figure 4.1, the expression SBEL(e) holds. For example, SBEL(E_3) and SBEL(E_4) are both *True*. The intersection between H_S and S contains the events which are (supposed to be) believed by both, whereas the set $H_S \setminus S$ constitutes the *conflictual* set, containing events believed by the hearer but not by the speaker.

In order to allow the language *Lo* to represent this situation, it is necessary to introduce a modal operator

HBEL(x)

which represents the proposition "the Hearer BELieves x". Hence, H_S, in Fig. 4.1, can be defined as the set containing all and only the events 'e' such that HBEL(e).

The graphical representation of Figure 4.1 underlines that the *real* set of events believed by the audience, indicated by H, may differ from the one assumed by the system. In particular, there might be events, such as E_1, which are not part of the hearer's beliefs, while the speaker supposes they are, or like E_2, which the hearer believes despite the speaker's expectations.

It should be noticed that the specific syntax for the set of expressions qualified as 'events' has not been given yet; however, one of their main characteristics consists of the fact that that they can be *negated.* Let the expression

NOT(e)

indicate the *negation* of the event 'e': NOT(e) can be defined as the event which happens *iff* 'e' does not happen. Obviously, two nested negations annul each other:

$$\text{NOT}(\text{NOT}(e)) = e \tag{4.1}$$

The proposition corresponding to the event NOT(e) can be obtained simply by negating the *action* of e. For example, if E_1="*John likes Mary*", then the negated event NOT(E_1) will be "*John does not like Mary*".

Nevertheless, in many cases, the common meaning of 'negation' in natural language does not correspond exactly to the definition given, because of the different *relevance* which humans normally assign to different components of an event.

For example, let E_1 be the event "Bill met his girlfriend for lunch, today". If the answer (of a human agent) to the question

"Did Bill meet his girlfriend for lunch, today?"

is "No, he did not", and if we represent this answer with NOT(E_1), then there can be at least two different interpretations for the expression NOT(E_1): 1) Bill did not meet his girlfriend today, *at all*; or 2) Bill did not meet her *for lunch*, but he might have met her for another occasion, during the day.

Although the definition of negation of an event given above should always be interpreted as in the latter case, the former possibility might be the most common one in natural language. This is because, in this specific example, the relevance of

the feature *occasion* (lunch) might be considered small if compared with the 'main' element of the event, namely, to meet a girlfriend. Thus, the simple assertion — in natural language — of NOT(E_1), deprived of any further detail (such as "No, he met her for dinner"), will probably be interpreted as negating the most relevant element of the event, i.e. the meeting itself.

Nevertheless, in order for the system to be able to distinguish between these two cases, it would be necessary to effect a *semantic* analysis of the content of the events, taking into account the different relevance of the elements in connection with the current *context* of the discourse, a problem which goes beyond the declared scope of this work.

In conclusion, the term 'event' should be hereafter interpreted as a semantic unit which could also represent the negation of another event. In fact, the set $S \cup H_S$ of Fig. 4.1, containing all the speaker (and hearer) beliefs, might as well contain negative events, producing beliefs such as SBEL(NOT(e)) or HBEL(NOT(e)). Indeed, since NOT(E_4) belongs to H_S, it results HBEL(NOT(E_4)), whereas SBEL(E_4).

Another important requirement of the system consists of its ability to manage *uncertain* information. From a practical point of view, the presence of uncertainty is a fundamental aspect of the real world. To quote Clark: "Uncertainty is present in most tasks that require intelligent behaviour" ([22], p.109). The limited and subjectivity-prone resources of a human being require a system to be able to deal with uncertain information in most of the cases.

The specific model developed adopts a qualitative[2] *tripartite* approach, also used, among the others, by Ellis [33] and Gärdenfors [54]. In brief, the *attitude* of the system towards every event 'e' must be one (and one only) of the following three:

a) *Speaker believes* e;

b) *Speaker is undecided about* e;

c) *Speaker believes* NOT(e).

The attitude *a*) — SBEL(e) — is adopted by the system when the event 'e' is considered *True*, whereas case *c*) — SBEL(NOT(e)) — represents the opposite situation, in which the system believes 'e' to be *False*. The attitude *b*) of *uncertainty*

[2]See [22] for an interesting comparison between numerical (quantitative) and symbolic (qualitative) techniques for uncertainty management.

towards 'e' is held when there are not enough information for the system to decide whether 'e' should be evaluated *True* or *False*.[3] Notice that requiring that one of these attitudes be held towards an event corresponds to imposing a specific set of *restrictions* on the language. The expression of uncertainty is allowed by the introduction of the following modal operator in the outer language *Lo*:

SUND(e)

The possibility of uncertain attitudes in the speaker's beliefs requires the introduction of a symmetrical modal expression

HUND(e)

for the hearer model representation.

Although, at first glance, the three cases HBEL(e), HUND(e) and HBEL(NOT(e)) seem to be adequate to describe the hearer model, there is a category of events which would be difficult to classify under any of these attitudes. For example, suppose the speaker has a good knowledge of computer science, and believes in the event E_1="The computational complexity of any known algorithm which calculates the permutations of a set of objects grows exponentially with the number of objects". If the hearer does not have the same scientific background, (s)he is not likely to have any knowledge of computational complexity whatsoever. In fact, it may as well be that the hearer had never 'heard' this concept before, or not even 'conceived' it as a thought.

What attitude should a hearer be expected to have towards an event which (s)he has never considered before? The hearer could not possibly believe, disbelieve or even be uncertain towards something over which (s)he has never pondered. The last modal operator of *Lo*, introduced to represent this situation, is

HUNK(x)

which should be read as "the event x is *UNKnown* to the Hearer", meaning that the concept x has never been conceived as a thought in the hearer's mind. The reciprocal notion of this concept (namely, 'KNOW-ABOUT') has been adopted by Moore and Paris in [98, p.671].

The symmetrical form for the speaker, namely SUNK(x), is also a correct expression of *Lo*, although it will never be possible for an event in the speaker's

[3]The simple tri-partition can be further refined into many degrees of belief, as shown, for example, in [42].

knowledge to be unknown to the speaker. In fact, as soon as the speaker simply *thinks* about the concept SUNK(x), this becomes immediately false by definition, as the event x is not anymore *unknown*. This situation presents an interesting parallelism with the Cartesian basic belief "I think", which, analogously, becomes immediately true as soon as the speaker conceives it.

Notice that this attitude should not be confused with a situation of *incomplete* knowledge: for example, the speaker might believe that (s)he does not know at what time the next train to London will leave, but the event "The next train to London will leave at a *certain* time" has been conceived as a thought, and, even though not complete in its details, is not an unknown event[4].

In fact, it would be possible to extend the representation by introducing a notation for events having components which are only *partially* known by the speaker, but believed to be known by the hearer. However, this kind of situations can also be captured by using the verb '*to know*' (or its negation) as the *action* of the events. In the previous example, we could have E_1="The hearer *knows* at what time the next train to London will leave", and SBEL(E_1).

If we allow events as subject or objects of other events and consider all of the previous predicates as representing a specific kind of *events*, then the language becomes much more sophisticated, and permits the construction of *mutual* belief expressions (e.g. SBEL(HBEL(SBEL(x)))), *nested* beliefs, such as SBEL(SBEL(SBEL(x))), or more complex expressions, like SBEL(NOT(HBEL(x))) or NOT(HUND(SUND(x))).

Moreover, this assumption allows the hypothetical model of the hearer to be simply considered as part of the speaker model: *any* expression, including those describing the hearer model (such as HBEL( ) or HUNK( )), will be an element of the set S of events *believed* by the speaker.

Before defining precisely the final syntax of these expressions, it is necessary to decide *how many levels* of nested predicates the language *Lo* should allow.

Since this language is intended to be used by the speaker to build a persuasive discourse, it must be able to represent *disagreement*, i.e. situations in which the hearer's (expected) belief might be different from the speaker's one. For example, SBEL(E_1), but SBEL(HBEL(NOT(E_1))).

[4]The difference between these two meanings of the verb *to know* is similar to the distinction that, in German, is marked by the two verbs *kennen* and *wissen*, the first one being used mostly to express the idea of knowing a concept as being 'acquainted' to it (see also [1, p.61]).

Moreover, in many cases of arguments, the conflictual situation between the two (or more) agents is due to a basic 'misunderstanding' in which an agent believes the other(s) to have a certain belief, whereas this is not the case. For example, in the speaker model, we could have HBEL(SBEL(E_2))[5], whilst it is true that SBEL(NOT(E_2)).

In order for the speaker to be able to deal with and solve such a situation, it seems appropriate to include this kind of expressions in the language *Lo*, and endow the system with a — hypothetical — model of the hearer's model of the speaker's beliefs.

A further level of nesting in the expressions would lead to beliefs such as HBEL(SBEL(HBEL( ))), which represent (from the point of view of the system) the hearer's view of the of the speaker's model of the hearer's beliefs. This sort of hypothetical model, besides being rather complex and unusual in common situations, requires also a notable amount of knowledge and assumptions about the hearer's mental state, which are likely to be collected only in specific contexts, such as protracted dialogues or prolonged shared experiences.

Furthermore, a three-level nested belief expression is often associated with an attempt to produce (or resist) *deception*: according to Cohen and Perrault,

> "...if AGT1 successfully lied to AGT2, he would have to be able to believe some proposition 'p', while believing that AGT2 believes that AGT1 believes p is false (i.e. AGT1 BELIEVE AGT2 BELIEVE AGT1 BELIEVE ($\sim p$))." [25, p.183]

Since this model is meant to produce persuasive discourses without using deceptive techniques (as specified by the *sincerity axioms*), the syntax for the outer language *Lo* has been restricted to only two levels of adjacent modality nesting.

The first half of the BNF formalization of the grammar $<WFE>$ ('Well Formed Expressions') generating the set of correct query expressions contained in *Lo* is given below.

$$
\begin{aligned}
<WFE> &::= <P1> \mid <P2> \mid <A> \\
<P1> &::= <Pred>(<A>) \mid \text{NOT}(<Pred>(<A>)) \\
<P2> &::= <Pred>(<P1>) \mid \text{NOT}(<Pred>(<P1>)) \\
<Pred> &::= \text{SBEL} \mid \text{SUND} \mid \text{SUNK} \mid \text{HBEL} \mid \text{HUND} \mid \text{HUNK}
\end{aligned}
$$

[5]Notice that this expression should be thought of as having an 'SBEL' predicate in front of it.

$$<A> \;::=\; <E> \mid \text{NOT}(<E>)$$

This grammar contains all the expressions analysed before, and allows two levels of belief nesting (production rule $<P2>$). Each of the two levels can be negated, i.e. can have a 'NOT' in front of the predicate. Since every belief should be thought of as having the prefix 'SBEL' in front of it, the number of levels of nesting can actually be considered equal to three.

Since $<WFE>$ generates exclusively expressions obtained from $<P1>$, $<P2>$ and $<A>$, and each of these symbols can produce, in turn, a specific sub-expression or the *negation* of the same sub-expression, every event of *Lo* has its negation in *Lo*. For example, both of the expressions NOT(HBEL(SBEL(NOT(e)))) and corresponding negation HBEL(SBEL(NOT(e))) are generated by $<WFE>$.

The inner content of a predicate is defined by the production for the symbol $<A>$, or 'atom', which can be either an 'event' $<E>$ or its negation. The precise production rule for the symbol $<E>$ will be described in the next section.

The reader should bear in mind that the distinction made here between 'atoms', 'events' and 'predicates' is purely syntactical: all of these expressions should be considered as different kinds of *events*, as mentioned earlier in the section.

This first part of outer language allows to combine various predicates into the two levels of nesting of an expression. Even though the meaning of each predicate of $<Pred>$ has been clarified, the interpretation of their *combinations* in a two-level expression is yet to be explained.

In fact, for example, what is the meaning of the query SUND(NOT(HBEL(e)))? And, if this meaning may appear 'intuitive', what is the difference — if there is one — between the previous query and SUND(HBEL(NOT(e)))? In other words, on which basis can these more complex expressions be evaluated *True* or *False*?

The 'algorithmic' meaning of all of the outer expressions will be defined, subsequently, by the function of interpretation $\mathcal{I}(\,)$, which will identify the specific tests which will have to be carried out on the state in order to evaluate any possible query. However, the definition of $\mathcal{I}(\,)$ will be based on the 'intuitive' meaning of the expressions of $<WFE>$. This meaning, specified in terms of the *semantic* relations which hold between different *groups* of expressions, is described below.

The Meaning of the Attitudes

The outer language generated by $<WFE>$ contains expressions which are *semantically* equivalent. The elements of each pair of the following list have been considered as having the same meaning:

1) SUND(NOT(x)) $\Leftrightarrow$ SUND(x)
2) SUND(HBEL(x)) $\Leftrightarrow$ SUND(HUND(x))

3) HUND(NOT(x)) $\Leftrightarrow$ HUND(x)
4) HUND(SBEL(x)) $\Leftrightarrow$ HUND(SUND(x))

5) HBEL(HUND(x)) $\Leftrightarrow$ HUND(x)
6) HBEL(HBEL(x)) $\Leftrightarrow$ HBEL(x)

7) SBEL(SUND(x)) $\Leftrightarrow$ SUND(x)
8) SBEL(SBEL(x)) $\Leftrightarrow$ SBEL(x)

Equations 1) and 3) formalise the concept according to which if an agent is undecided about the truth of an event, (s)he must also be undecided about its negation. Notice that while the first equivalence concerns the speaker model, the other one is defined for the expressions specifying the hearer model.

Equivalence 2) associates SUND(HBEL(a)) with SUND(HUND(a)). The meaning of the first form can be thought to be "The speaker is undecided about whether the hearer believes or not the event a". Generalizing, this can be seen as being undecided about the *opinion* of the hearer towards the event 'a', opinion which could be either of belief, disbelief or uncertainty. Therefore, the three expressions

a. SUND(HBEL(a))

b. SUND(HBEL(NOT(a))

c. SUND(HUND(a))

can be considered semantically equivalent, and assumed to represent the same state of *uncertainty* of the speaker towards the hearer's opinion about 'a' [6].

[6]Notice that SUND(HBEL(NOT(a))) is equivalent to SUND(HUND(NOT(a))) because of 2); this expression, then, becomes SUND(HUND(a)) through 3).

Equivalence 4) is analogous to 2), if each 'S' is replaced with 'H' and vice versa: the three expressions

a'. HUND(SBEL(a))

b'. HUND(SBEL(NOT(a))

c'. HUND(SUND(a))

have to be considered equivalent.

Finally, equations 5)–6) and 7)–8) represent the implementation of the rules of positive and negative *introspection* (axioms 4 and 5 of modal logic) and 'subjective' knowledge (axiom T) for the speaker and hearer model, respectively (see page 25). They formalise the concept according to which, if an agent believes (or is undecided about) some event, then (s)he *believes* to be in that attitude, and vice versa.

Besides the previous relations of equivalence between syntactically different queries, the expressions of the language $<WFE>$ have been grouped into three semantic 'levels', and the expressions of each level have been required to satisfy a condition of *mutual exclusivity* and *exhaustivity*.

The three sets of expressions subject to this *restriction* are reported below:

$S_1(e)$ = { e, NOT(e), SUND(e) };

$S_2(e)$ = { HBEL(e), HBEL(NOT(e)), HUND(e), HUNK(e), SUND(HUND(e)), SUND(HUNK(e)) };

$S_3(e)$ = { HBEL(SBEL(NOT(e))), HBEL(SUND(e)), HBEL(SUNK(e)), HBEL(SBEL(e)), HUND(SUND(e)), HUNK(e), HUND(SUNK(e)) }.

with $e::<E>$.

In other words, for any given event $e::<E>$, the set S^* of queries (formulae of *Lo*) evaluated *True* by the belief system must satisfy the property according to which *one and only one* expression of each of the sets $S_1(e), S_2(e), S_3(e)$ is evaluated *True* [7].

[7] In order to guarantee that this property is always respected, a set of *restrictions* will have to be imposed on the inner language of the belief system.

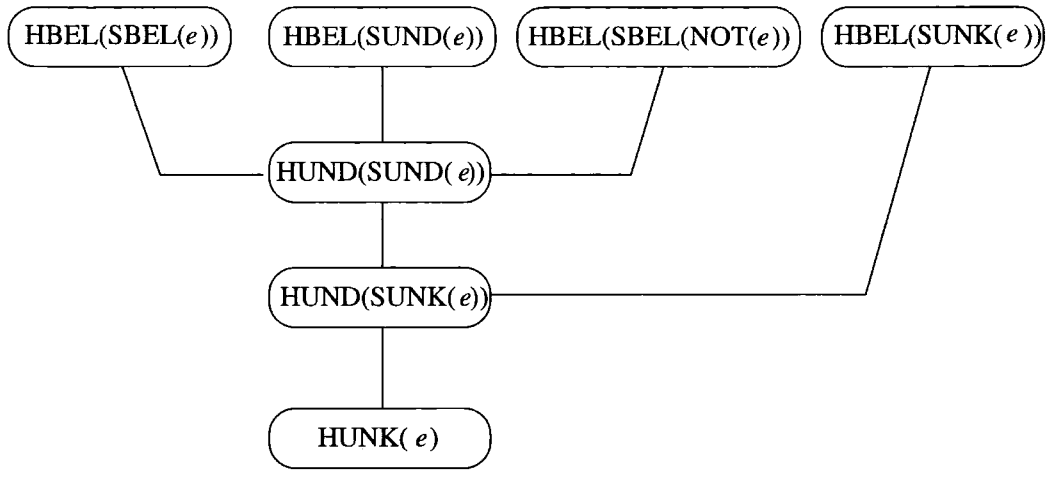


Figure 4.2: Hierarchical representation of the set $S_3(e)$ of mutually exclusive and exhaustive beliefs.

Notice that the expressions of S_1 represent the tri-partition of the speaker's attitudes already introduced at page 81; S_2 contains the attitudes which define the *model of the hearer*, whereas the set S_3 contains the expressions concerning the *hearer's model of the speaker.*

In order to clarify the meaning of the expressions contained in S_1–S_3, consider the diagram of Figure 4.2. The tree organizes hierarchically the content of $S_3(e)$, so that the higher the level in the tree, the more specific the knowledge that the hearer has regarding the speaker's opinion, and the lower the level of *uncertainty* of the hearer model. The lowest node (root) HUNK(e) corresponds to no knowledge at all with respect to e, whereas the four leaves at the top specify clearly the hearer model of the speaker's opinion towards e.

If an expression is placed at the intersection of branches, it represents an *uncertain* state, in which one of the opinions of the higher nodes connected *directly* to the intersection is supposed to hold, but it is not clear which one it is.

For example, the expression HUND(SUNK(e)) can be defined as representing the hearer's uncertainty between the two following cases: 1) HBEL(SUNK(e)) and 2) HUND(SUND(e)). In case 1, the speaker (in the hearer's model) does not *know* the event e. In case 2, the hearer believes the speaker to know e, but is uncertain about the *specific* attitude — amongst the three possible SBEL(e), SBEL(NOT(e)) and SUND(e) — that the speaker holds towards the event.

Swapping the prefixes 'S' with 'H' for each predicate present in the tree will produce a new diagram containing the analogous hierarchical structure for the set

$S_2(e)$ [8].

Consider now the expressions in the form HBEL(NOT(S...(a))): their interpretation can be deduced directly from the diagram of Figure 4.2.

For example, HBEL(NOT(SBEL(a))) means that the hearer believes that it is not the case that the speaker believes 'a'. Thus, the alternative opinions which are left are SBEL(NOT(a)), SUND(a) and SUNK(a) (as beliefs of the hearer's model), for if the hearer is *sure* that the speaker does not believe a, then it is not possible for the hearer to be in the state of uncertainty HUND(SUND(e)) — which would admit the case HBEL(SBEL(a)) — nor to be in the even more uncertain state HUND(SUNK(e)). Hence:

$$\text{HBEL(NOT(SBEL}(a))) \iff \text{HBEL(SBEL(NOT}(a))) \vee \text{HBEL(SUND}(a)) \vee \text{HBEL(SUNK}(a))$$

Similarly, for the other two possibilities, we have:

$$\text{HBEL(NOT(SUND}(a))) \iff \text{HBEL(SBEL(NOT}(a))) \vee \text{HBEL(SBEL}(a)) \vee \text{HBEL(SUNK}(a))$$

and

$$\text{HBEL(NOT(SUNK}(a))) \iff \text{HBEL(SBEL(NOT}(a))) \vee \text{HBEL(SBEL}(a)) \vee \text{HBEL(SUND}(a)) \vee \text{HUND(SUND}(a))$$

Belief by Default

The imposition of the restrictions on the sets S_1–S_3 can give rise to a problem of *finiteness* of the state representation.

In fact, consider the set of possible events generated by $<E>$. If the cardinality of this set is infinite[9], the number of *attitudes* to be represented is also infinite. This is due to the fact that, for every event $e :: <E>$, at least one of the three beliefs of the set $S_1(e) = \{ e, \text{NOT}(e), \text{SUND}(e) \}$ has been required to hold.

If we simply included in the state of the system the specific event of $S_1(e)$

[8]Notice that the new root, SUNK(e), should be dropped from the diagram, being one of the contradictory expressions. Moreover, in order to obtain the *exact* content of $S_2(e)$, any prefix 'SBEL' should also be removed from the new nodes.

[9]E.g., $<E>$ could be allowed to generate an infinite list of primitive events $E_1, E_2, \ldots$.

which is currently believed by the speaker, and repeated this operation for every possible $e ::< E >$, we would end up with a state containing an *infinite* number of propositions.

However, the mechanism of *interpretation*, introduced in the paradigm of the belief system, allows this problem to be easily solved through the adoption of a 'default' belief. The solution consists of assuming one of the possible attitudes to be the most common and considering it as the *default* attitude of the system. This attitude will be automatically assumed towards *any* event $e :: < E >$ — without any need to mention the event e in the state — iff *none* of the others attitudes is explicitly stated. The three default expressions chosen for S_1–S_3 are shown below:

a) SUND(e), for $S_1(e)$;

b) SUND(HUNK(e)), for $S_2(e)$;

c) HUND(SUNK(e)), for $S_3(e)$.

For example, given $e = \mathrm{E}_1$, the belief SUND(E_1) will be evaluated *True* by the system iff neither E_1 nor NOT(E_1) belongs to the current state. More formally,

$$\mathrm{SUND}(e)) \Longleftrightarrow \neg\,(\,B(e) \vee B(\mathrm{NOT}(e))\,)$$

The implementation of the three default mechanisms — and also of the expressions in the form HBEL(NOT(S... (a))) considered before — will be realized through the *function of interpretation* $\mathcal{I}(\,)$, which can define the truth of any outer query as a *boolean function* of (inner) terms. For example, for $S_1(e)$,

$$\mathcal{I}(\mathrm{SUND}(e)) = \neg\,(B(e) \vee B(\mathrm{NOT}(e)))$$

The complete specification of the interpretation $\mathcal{I}(\,)$ will be given in Section 4.2.2.

Features

Whilst the syntax for the predicates, described in the previous section, is intended to capture the different *attitudes* which speaker and hearer have towards a set of events, the second group of expressions concerns the representation of the *features* of an event and of the relations between different events. Such features have been introduced to render 'explicit' the *relevant* information encapsulated in the atomic units of 'events', and normally not accessible (see page 77); such information are useful for the construction of the discourse structure and for the representation of the structure of the belief system itself.

In logic, given a proposition p and a formal theory, if p is a *theorem* of that theory, then p must be either a consequence of inference rules and other *theorems*, or be one of the axioms of the theory. Similarly, in a rational system, the presence of a belief must be *justified* (see Sections 2.2.1 and 2.2.2). According to the 'foundations' approach, if an agent believes an event 'e' to be *True*, then 'e' must either be a consequence of a process of *reasoning* based on other beliefs, or constitute a *grounded* belief, i.e. an event whose truth is, from the agent's point of view, unquestionable.

From a 'cynical' point of view, shared by Armstrong [8], the only beliefs in our mind which we do not normally question are those deriving from our direct experience of the physical reality, that is from our *sensorial perception* of the external world. For example, if the agent A experiences the event E_1="Agent A sees a table", then A will strongly believe E_1.

However, considering that the belief system is designed to be used for discourse generation purposes, a speaker's belief will be allowed to be left 'unjustified' when it is already (supposed to be) believed by the *audience*. In other words, if the event 'e' is (for the speaker) fully justified — through a reasoning process — *and* HBEL(e) holds, then there is no need to represent, in the knowledge of the belief system, the justification(s) for 'e', as this 'supporting evidence' will not have to be mentioned in the discourse[10]. This idea is in accordance with the 'coherence' approach, introduced in Sections 2.2.1 and 2.2.2.

If the event 'e' is not believed by the audience and the discourse aims require the hearer to believe it, it will be necessary, for the discourse to be effective, to use the persuasive action of the events which, in the speaker's view, justify the validity of the belief. As Cohen and Perrault pointed out,

> "...before AGT will be convinced, she needs to know the justifications for AGT1's belief, which may require that AGT believes (or be CONVINCED of) the justifications for believing those justifications, etc." [25, p.193] [11]

Notice that the grounding of events through information provided by other sources (i.e. directly by human agents, or through *media*, such as books, newspapers, etc.) is a special case of the grounding through sensory experience, although the credi-

[10]This consideration is based on the assumption that model does not use rhetorical forms.
[11]In this context, AGT and AGT1 represent, respectively, the hearer and the speaker.

bility of the information provided will have to be based also on the *reliability* of the source, which, in turn, might require further justification (see [123] for a similar discussion on belief grounding).

In order to represent the mentioned aspects of belief justification — namely, reasoning and sensory input — and the relevant relations between events (e.g. support and attacks) which are useful for the discourse structure construction, the following expressions have been included in *Lo*:

a) relation of *support* between events: SUP(e, e');

b) Speaker or Hearer *Real* events: SR(e), HR(e');

c) Speaker or Hearer *Inductive EXPeriences*: SIEXP(e), HIEXP(e);

More precisely, these expressions constitute the different forms which an event generated by $<E>$ can assume. Let us examine these new terms of the outer language *Lo*.

a) If the belief in e' is a consequence of reasoning, then there must be (at least) another event e such that the belief in e supports the belief in e'. The expression SUP(e, e') means that the belief in the event e is a *sufficient* reason to believe e', but it does not imply that either e or e' have to be believed.

Notice that SUP(e, e') constitutes an event itself, and, in order to be believed, it should be justified. Hence, we could have supporting evidence for a 'support' event, e.g. SUP(e,SUP(e', e'')). Notice also that this event could be believed by the speaker, but not by the hearer, or vice versa. This underlines the fact that the model allows *subjective* interpretations of the world: different agents might have different perceptions of the idea of 'sufficient justification' for a specific event, and thus have different opinions towards a SUP( ) event.

b) The expressions SR(e) and HR(e) indicate that the event e is a *real* experience for, respectively, the speaker and the hearer. In other words, the agent believes e because (s)he has perceived its happening through *sensorial* experience. This means that the agent was involved directly in the event as its *object* or *subject*, or simply was a 'spectator' of the event.

Even in this case, the expression SR(e) constitutes an event itself, but its presence in the set of the speaker's beliefs does not require any further justification or

support, neither does the presence of the event 'e'. The same property applies to HR(e), in the (speaker's) model of the hearer. For example, if HR(E_3) is assumed to hold, then no further support is needed, in the hearer model, to justify the presence of the belief HBEL(E_3)[12].

Notice that if e=SUP(E_1,E_2), the expression SR(e) should not be considered correct. In fact, a *support* relation cannot be 'experienced' physically: what can be *perceived* is simply the happening of E_1 and the *subsequent* happening of E_2.

The only way in which a support relation between two events E_1 and E_2 can be learned through sensorial perception consists of a sequence of *repeated* experiences in which, every time, the happening of E_1 produces the happening of E_2. In this case, the mechanism of *induction* will lead to a generalization, producing the ground to believe that a causal relation between the two events does exist.

c) In order to distinguish a 'normal' real experience from an *inductive* one, the two following additional event-qualifiers have been included in the language *Lo*:

SIEXP(s), HIEXP(s)

indicating, for the speaker and hearer, that the support event s=SUP( ) is an Inductive EXPerience of the agent. The properties and characteristics of these expressions (e.g. no need for further justification) are the same as those of the corresponding forms SR( ) and HR( ).

As mentioned in the previous sub-section, for every event e generated by $<E>$ there are three possible attitudes which the system-speaker can adopt: SBEL(e), SUND(e) and SBEL(NOT(e)). Since *at least one* of these attitudes must be chosen, the decision about which category an event e will belong to should be taken by assessing the various supports which hold in favour of e and in favour of NOT(e).

For example, if neither e nor NOT(e) has any *supporting* argument — i.e. another event e' such that SBEL(e') and SBEL(SUP(e', e)) or SBEL(SUP(e',NOT(e))) — then the belief SUND(e) should be adopted by the system.

Another situation in which an uncertain attitude should be chosen is represented

[12]Since HR(E_3) is one of the speaker's beliefs, it should be justified. However, this would not be necessary if, for each belief HR(e), the model also assumed HBEL(HR(e)): in fact, this would allow HR(e) to be left 'unjustified', as already believed by the audience. This assumption will actually be adopted by the model through one of its *inference rules* (Section 4.2.2).

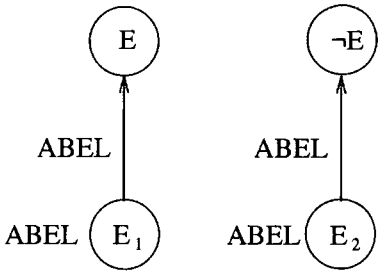


Figure 4.3: Graphical representation of balanced supports for E and NOT(E).

in Fig. 4.3, in which two contradictory events have, simultaneously, supporting arguments.

The relation SUP(e, e') should not be treated simply as the logical implication, $e \rightarrow e'$ (read "e *implies* e'"), but, rather, as a 'defeasible' form of implication. This peculiar interpretation given to the SUP relationship is originated by the presence of *uncertainty* in the model. In fact, it is possible to show that *only if* the possibility of an uncertain attitude is *excluded*, does believing SUP(y, x) entail believing SUP(NOT(x),NOT(y)).

To see this, assume a generic agent A to believe the two following events:

NOT(x), SUP(y, x)

Let us first consider the case in which there can be *no* uncertainty in the model, which means that A must *necessarily* believe either y or NOT(y), that is $y \vee$ NOT(y) must be always *True*.

If ABEL(y)[13], then there would be supporting evidence for A to believe x, which contradicts ABEL(NOT(x)). Hence, for the law of the *excluded middle*, it must be ABEL(NOT(y)). In this case, believing SUP(y, x) and NOT(x) has led to NOT(y), as if SUP(y, x) were equivalent to SUP(NOT(x),NOT(y)).

Nevertheless, in a model which admits uncertainty, the fact that A believes NOT$(x) \wedge$ SUP(y, x) leads only to the conclusion that A cannot — coherently — believe y, but leaves open two possible attitudes: ABEL(NOT(y)) and AUND(y)[14]. Hence, in this case, it is not possible to conclude ABEL(NOT(y)) by exclusion, and the SUP( ) relationship cannot be interpreted exactly as the logical implication.

Summarizing, the second half of the BNF formalization of the grammar $< WFE >$, generating the outer language *Lo*, is given below:

[13]Read "agent A BELieves y".
[14]Read "A is UNDecided about y".

$$
\begin{aligned}
<E> \;&::=\; \mathrm{SR}(<Ei>) \mid \mathrm{HR}(<Ei>) \mid \mathrm{SUP}(<A>,<A>) \mid \\
&\quad\; \mathrm{SIEXP}(<Su>) \mid \mathrm{HIEXP}(<Su>) \mid <Ei> \\
<Su> \;&::=\; \mathrm{SUP}(<A>,<A>) \mid \mathrm{NOT}(\mathrm{SUP}(<A>,<A>)) \\
<Ei> \;&::=\; \mathrm{E}_0 \mid \mathrm{E}_1 \mid \mathrm{E}_2 \mid \mathrm{E}_3 \mid \ldots \mid \mathrm{E}_n \ldots
\end{aligned}
$$

The symbol $<A>$ corresponds to $<E> \mid \mathrm{NOT}<E>$, as already introduced in the first part of the BNF formalization, given in the previous section.

It should be noticed that the inner argument of the two 'real' expressions SR( ) and HR( ) cannot be a *negative* event. This is because it is not possible to experience something which is *not*. Although in natural language there are expressions which could be interpreted as such, their implicit meaning, after a careful analysis, appears to 'summarize' a collection of positive facts. This view is also endorsed by Armstrong, who writes:

> "It seems reasonable to say instead that the nature of a thing is exhausted by its *positive* properties. That is to say, positive properties are a thing's only properties. It is they, in all their multitude and diversity, which make negative predicates applicable to objects." [8, p.124]

For example, consider the event "My watch does *not* work": although it would appear to be a real *negative* experience, it consists actually of the result of the *repeated* (positive) real experiences "Its hands are completely immobile" and "It is completely silent", collected continuously during the last minute or so.

On the other hand, the argument of an *inductive* real experience *can* be a negated event. For example, the expression $\mathrm{SIEXP}(\mathrm{NOT}(\mathrm{SUP}(\mathrm{E}_1,\mathrm{E}_2)))$ is perfectly acceptable. In fact, this event represents *one* or many counter-example experiences, in which the realization of the event E_1 did not lead to the happening of E_2.

Notice also that each of the two arguments of a SUP( ) event is allowed to be an 'atom' $<A>$, which might be replaced, in turn, by another SUP( ) event, and so forth, yielding to expressions with an arbitrarily-high level of nested supports.

Finally, a *belief* event cannot be considered a real experience. If that were the case, then the belief would not need further support, whereas we expect the model to have a rational justification for each belief which is not grounded. In fact, a real experience event can only be an event involving the agent's sensorial perception, and the only expressions allowed to represent it are the 'primitive'

events $E_0, E_1, E_2, \ldots$.

Shown below are few examples of query expressions generated by $<WFE>$:

NOT(HBEL(NOT(HBEL(E_1))))
HUND(NOT(SR(E_{34})))
HBEL(NOT(SUNK(SUP(E_4,NOT(E_3)))))
NOT(SUND(HUNK(SUP(HR(E_2),SUP(E_3,E_5)))))

One of the possible derivations of the first query is reported below:

$<WFE> \rightarrow <P2> \rightarrow$ NOT($<Pred>(<p1>)) \rightarrow$
$\rightarrow$ NOT(HBEL($<P1>$)) $\rightarrow$
$\rightarrow$ NOT(HBEL(NOT($<Pred>(<A>)$))) $\rightarrow$
$\rightarrow$ NOT(HBEL(NOT(HBEL($<A>$)))) $\rightarrow$
$\rightarrow$ NOT(HBEL(NOT(HBEL($<E>$)))) $\rightarrow$
$\rightarrow$ NOT(HBEL(NOT(HBEL($<Ei>$)))) $\rightarrow$
$\rightarrow$ NOT(HBEL(NOT(HBEL(E_1))))

The precise 'algorithmic' meaning of each outer expression generated by $<WFE>$ is determined by the specific function of *interpretation* $\mathcal{I}(\,)$, which is described in the following section.

4.2.2 The BP Algorithm 'A'

This section describes the main component of the belief system (Lo, Li, A), namely, the BP algorithm 'A'.

As explained in Section 3.2 and 3.2, the BP algorithm calculates the truth of every expression $w \in Lo$ as the logical value assumed by the boolean function $\mathcal{I}(w)$ in the 'completed' state $S' = \mathbf{C}_R(S)$, where S is the current state, and R the set of inference rules adopted.

The BP algorithm A can be summarized by the following abstract steps:

A

- let $w \in Lo$ be the expression to be evaluated, $S \subset Li$ the current state, R a set of inference rules on Li and $\mathcal{I}(\,)$ an interpretation of Lo on Li.

1 Calculate $S' = \mathbf{C}_R(S)$;

2 result:= $\mathcal{I}(w) \mid_{S'}$.

There are two important points which should be made clear.

First of all, there are expressions of *Lo* which should be always evaluated *True*. For example, as mentioned before, the event SUNK(x) cannot be held as one of the speaker's beliefs, and therefore its negation NOT(SUNK(x)) will always be evaluated *True*.

This can be realized by associating these expressions with the boolean function *constantly* true $\mathcal{F}_T \equiv True$, and associating their *negation* with the boolean function constantly false $\mathcal{F}_F \equiv False$.

The set of contradictory expressions, associated with $\mathcal{F}_F$, is reported below:

I) SUND(SR(e))	I') HUND(HR(e))
II) SUND(SIEXP(e))	II') HUND(HIEXP(e))
III) SUNK(x)	III') HBEL(HUNK(x))
IV) SUND(SUNK(a))	IV') HUND(HUNK(a))
V) SUND(SUND(a))	V') HUND(HUND(a))
VI) SUND(SBEL(a))	VI') HUND(HBEL(a))

The first two pairs exclude the possibility for an agent (speaker or hearer) to be undecided about its own real (or inductive) experiences: this means that, for any given event e::$<E>$, the agent knows exactly if 'e' does or does not constitute a real experience[15].

The two pairs III)–III'), IV)–IV') concern the concept of 'unknown': as already discussed, in the speaker model, the belief SUNK(x) is *False* for any expression x, and so is the symmetrical event for the hearer model HBEL(HUNK(x)). Consequently, it is not possible for an agent to be undecided about the *knowledge* of

[15]This excludes the possibility of uncertainty in the *memories* concerning the real experiences of an agent.

an event: as soon as the event 'a' is conceived, it becomes immediately known (IV–IV').

The last two pairs exclude states of uncertainty of an agent towards its own opinions concerning an event $a::<A>$: an agent knows whether or not (s)he believes (or is undecided about) a specific event.

The set of *tautologies* is obtained simply by negating all the contradictory expressions, that is by adding the prefix NOT( ) to each term of the previous list of pairs. Moreover, using the pairs I)–I'), II)–II'), other four tautologies can be identified, consisting of the same negated expressions to which the prefix HBEL( ) has been added: this is equivalent to 'transfer' these tautologies into the hearer model[16].

The second important observation concerns the characteristics of the interpretation function $\mathcal{I}(\,)$: if the expression $w \in Lo$ belongs also to the *inner* language $Li \subseteq Lo$, the function associated to w is simply $B(w)$ (by definition of interpretation; see also Equation 3.2), where $B(\,)$ is the boolean evaluation of Li determined by the 'completed' state S'. This means that $\mathcal{I}(\,)$ must be such that

$$\forall w \in Li, \qquad \mathcal{I}(w) \mid_{S'} = \begin{cases} True & \text{if } w \in S' \\ False & \text{otherwise.} \end{cases}$$

In order to define entirely the algorithm A, let us begin by specifying the interpretation $\mathcal{I}(\,)$.

Interpretation

The interpretation $\mathcal{I}(\,)$ associates every expression of Lo with a boolean function defined over a subset of Lo. The specific interpretation adopted by the BP algorithm 'A' is described below.

Let w be an expression of $Lo \leftarrow <WFE>$; the boolean function $\mathcal{I}(w)$ associated to w is

$$\mathcal{I}(w) = check(w') \tag{4.2}$$

where w' and the function $check(\,)$ are defined as follows:

1. w' is obtained from w by applying, whenever possible, the following syntac-

[16]Notice that the same operation could be done for the other tautologies, but the resulting expressions would not belong to the language $<WFE>$, as containing three levels of nested predicates.

tical transformations:

0. NOT(NOT(a)) $\longrightarrow a$

1. SUND(NOT(x)) $\longrightarrow$ SUND(x)
2. SUND(HBEL(a)) $\longrightarrow$ SUND(HUND(a))

3. HUND(NOT(x)) $\longrightarrow$ HUND(x)
4. HUND(SBEL(x)) $\longrightarrow$ HUND(SUND(x))

5. HBEL(HUND(x)) $\longrightarrow$ HUND(x)
6. HBEL(HBEL(x)) $\longrightarrow$ HBEL(x)

7. SUNK($p(a)$) $\longrightarrow$ SUNK(a)
8. HUNK($p(a)$) $\longrightarrow$ HUNK(a)

for any string x and any *one*-argument expression $p(a)$ such that the left hand sides of the reductions are expressions of *Lo*. For example, SUNK(NOT(a)) becomes SUNK(a) (using 7), but SUNK(SUP(e, e')) will be left unchanged, as $p(a)$=SUP(e, e') has two arguments.

It is important to notice that these simplifications should be applied to *every part* of w (i.e. even inside other predicates) and until there are no more transformations applicable. For example, the expression HBEL(HUND(NOT(a))) could be firstly reduced to HBEL(HUND(a)) using reduction 3, and then to HUND(a) via reduction 5.

2. The function *check*( ) is defined through the mechanism of 'pattern matching'. The expression w' is matched against a sequence of expressions — in a specific order — which might contain *variables*. A match returns *True* iff, for each variable v_i of the expression, there exists a string s_i such that, substituting every variable v_i with the associated string s_i, the two expressions result to be identical. It should be pointed out that the order in which w' is matched against the list of possible cases is important and needs to be respected to guarantee correct results. This is due to the fact that many of the case definitions of the function *check* are based on the assumption that the expression w' has not matched *any* of the previous possibilities, and therefore must be of a certain type.

The ordered sequence of matches is given below. Notice that the symbol '$\cong$' is used as abbreviation for the word 'matches', the symbol '_' represents a variable whose value is not relevant, and the notation '::' indicates the *type* of a variable, i.e. the production rule of $<WFE>$ which generates all the possible strings which can be used to instantiate the variable (e.g. if $a :: <A>$, then the variable 'a' can be instantiated to any string generated by $<A>$).

$check(w') =$

$\mathcal{F}_T$ if w' matches one of the *tautologies*;
$\mathcal{F}_F$ if w' matches one of the *contradictions*;
$check(x)$ if $w' \cong \text{SBEL}(x)$;
$\neg\, B(\text{SR}(e_i))$ if $w' \cong \text{NOT}(\text{SR}(e_i))$;
$\neg\, B(\text{SIEXP}(s))$ if $w' \cong \text{NOT}(\text{SIEXP}(s))$;
$B(w')$ if $w' :: <A>$;
$\neg\, check(x)$ if $w' \cong \text{NOT}(x)$;
$\neg\,(\ B(\text{HUND}(e)) \vee B(\text{HBEL}(\text{NOT}(e))) \vee B(\text{SUND}(\text{HBEL}(e))) \vee$
$B(\text{HUND}(e)) \vee check(\text{HUNK}(e))\ )$ if $w' \cong \text{SUND}(\text{HUNK}(e))$;
$B(w')$ if $w' \cong \text{SUND}(\text{HUND}(\_))$;
$\neg\,(\ B(e) \vee B(\text{NOT}(e))\ )$ if $w' \cong \text{SUND}(e)$;
$\mathcal{I}\,(\text{HUNK}(a)) \vee \mathcal{I}\,(\text{HUNK}(b))$ if $w' \cong \text{HUNK}(\text{SUP}(a, b))$;
$B(w')$ if $w' \cong \text{HUNK}(e_i)$;
$\neg\,(\ B(\text{HBEL}(\text{SBEL}(\text{NOT}(e)))) \vee B(\text{HBEL}(\text{SBEL}(e))) \vee B(\text{HBEL}(\text{SUND}(e))) \vee$
$B(\text{HUND}(\text{SUND}(e))) \vee check(\text{HBEL}(\text{SUNK}(e))) \vee check(\text{HUNK}(e))\ )$
if $w' \cong \text{HUND}(\text{SUNK}(e))$;
$B(w')$ if $w' \cong \text{HUND}(\_)$;
$B(\text{HBEL}(\text{SBEL}(\text{NOT}(e)))) \vee B(\text{HBEL}(\text{SBEL}(e))) \vee check(\text{HBEL}(\text{SUNK}(e)))$
if $w' \cong \text{HBEL}(\text{NOT}(\text{SUND}(e)))$;
$\mathcal{I}\,(\text{HBEL}(\text{SBEL}(\text{NOT}(a)))) \vee \mathcal{I}\,(\text{HBEL}(\text{SUND}(a))) \vee \mathcal{I}\,(\text{HBEL}(\text{SUNK}(a)))$
if $w' \cong \text{HBEL}(\text{NOT}(\text{SBEL}(a)))$;
$B(\text{HBEL}(\text{SBEL}(\text{NOT}(e)))) \vee B(\text{HBEL}(\text{SBEL}(e))) \vee B(\text{HBEL}(\text{SUND}(e))) \vee$
$B(\text{HUND}(\text{SUND}(e)))$ if $w' \cong \text{HBEL}(\text{NOT}(\text{SUNK}(e)))$;
$\neg\, check(\text{HUNK}(e))$ if $w' \cong \text{HBEL}(\text{NOT}(\text{HUNK}(e)))$;
$\mathcal{I}\,(\text{HBEL}(\text{NOT}(a))) \vee \mathcal{I}\,(\text{HUND}(a))$ if $w' \cong \text{HBEL}(\text{NOT}(\text{HBEL}(a)))$;
$B(\text{HBEL}(\text{NOT}(e))) \vee B(\text{HBEL}(e))$ if $w' \cong \text{HBEL}(\text{NOT}(\text{HUND}(e)))$;
$\mathcal{I}\,(\text{HBEL}(\text{SUNK}(a))) \vee \mathcal{I}\,(\text{HBEL}(\text{SUNK}(b)))$
if $w' \cong \text{HBEL}(\text{SUNK}(\text{SUP}(a, b)))$;

$B(w')$ if $w' \cong \text{HBEL}(\_)$;

If the type of a variable is not indicated, the variable can be matched with *any* expression of $<WFE>$.

Notice that the negation '$\neg$' indicates the unary logical operator, whilst the predicate 'NOT' is part of the syntax; for example, the expression $B(\text{NOT}(e))$ returns *True* iff the string "NOT(e)" is present in the current state, whereas '$\neg check(x)$' returns the boolean negation of the truth value of $check(x)$.

Summarizing, the second step of the algorithm A has been refined into the two following sub-steps:

2.1 $w' = simplify(w)$;

2.2 result:= $check(w')\ |_{S'}$.

Before analysing the reasons which motivated the previous definitions of the function *check*( ) and the simplifications 0–8, let us see, with an example, how they work.

Example 4.1 Suppose the expression which has to be evaluated to be

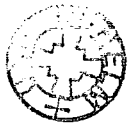
$$w = \text{SBEL}(\text{NOT}(\text{SUND}(\text{NOT}(\text{E}_1))))$$

First of all, simplification 1 will transform w into

$$w' = \text{SBEL}(\text{NOT}(\text{SUND}(\text{E}_1)))$$

Assuming that neither w' nor any of its sub-expressions — $\text{NOT}(\text{SUND}(\text{E}_1))$, $\text{SUND}(\text{E}_1)$ and E_1 — match a tautology or a contradiction, the boolean function $\mathcal{I}(w)$ associated to w can be deduced from the definition of *check*( ) in the following way:

$$\begin{aligned}
\mathcal{I}(w) &= check(w') = \\
&= check(\text{SBEL}(x)) && \text{– with } x = \text{NOT}(\text{SUND}(\text{E}_1)) \\
&= check(x) && \text{– by def. of } check(\,) \\
&= check(\text{NOT}(x')) && \text{– with } x' = \text{SUND}(\text{E}_1) \\
&= \neg\, check(x') && \text{– def. of } check(\,) \ (x \not:< A >)
\end{aligned}$$



$$
\begin{aligned}
&= \neg\, check(\mathrm{SUND}(\mathrm{E}_1)) && \text{- substitution}\\
&= \neg\neg(\, B(\mathrm{E}_1) \vee B(\mathrm{NOT}(\mathrm{E}_1))\,) && \text{- def. of } check(\,)\\
&= B(\mathrm{E}_1) \vee B(\mathrm{NOT}(\mathrm{E}_1)) && \text{- '}\neg\text{' elimination}
\end{aligned}
$$

In conclusion, $\mathcal{I}(w) = B(\mathrm{E}_1) \vee B(\mathrm{NOT}(\mathrm{E}_1))$. Hence, the expression w, which can be read as "the speaker believes not to be undecided about the event $\mathrm{NOT}(\mathrm{E}_1)$", will be evaluated *True* if either E_1 or $\mathrm{NOT}(\mathrm{E}_1)$ belongs to the (completed) state S'. In fact, if one of these two beliefs holds, then the speaker cannot be undecided about $\mathrm{NOT}(\mathrm{E}_1)$, as (s)he either believes it, or believes its negation.

Let us now examine the meaning of the nine simplifications applied to the expression w given in input to the BP.

The first one eliminates any double negation present in the event, according to Equation 4.1.

Simplifications 1–6 have been deduced from the relations of *equivalence* 1)–6) which hold between these expressions (see page 86). Their role consists simply of choosing one of the two forms as the 'standard' one. For example, because of equivalence 1), the two expressions

$$\mathrm{SUND}(a),\ \mathrm{SUND}(\mathrm{NOT}(a))$$

are considered *semantically* equivalent. Through simplification 1, the simpler non-negative syntax of the former has been chosen as representative of both: every occurrence of the latter form will be automatically reduced to this one.

It should be noticed that equivalences 7)–8) do not have a corresponding pair of simplifications: in fact, these relations, expressing the rule of reflection for the speaker, have been implemented directly through the definition of the function *check*( ), as it will be explained subsequently.

The last two simplifications are concerned with the concept of 'unknown' event. The idea behind them is that an event is unknown if it contains an unknown argument. For example, the event $\mathrm{HUND}(\mathrm{E}_1)$ cannot have been conceived in the hearer's mind if (s)he has never conceived the argument 'E_1'; neither could, for the same reason, the event $\mathrm{HBEL}(\mathrm{NOT}(\mathrm{E}_1))$, etc.

Hence, any query in the form HUNK( ) or SUNK( ) will be reduced to $\mathrm{HUNK}(e_i)$ and $\mathrm{SUNK}(e_i)$, where e_i ::<Ei> is the *innermost* argument of the initial ex-

pression. The only exception to this rule occurs when the expression contains a SUP( ) relation, and thus presents more than one innermost argument. In fact, the query HUNK(SUP(E_1,E_2)), for example, cannot be simplified, as the expression HUNK(E_1,E_2) would not be syntactically correct (as it cannot be generated by $<WFE>$).

However, the event SUP(E_1,E_2) should be considered unknown by an agent iff any of the events E_1, E_2 is unknown. More formally:

$$\text{HUNK}(\text{SUP}(a,b)) \iff \text{HUNK}(a) \vee \text{HUNK}(b) \tag{4.3}$$

$$\text{SUNK}(\text{SUP}(a,b)) \iff \text{SUNK}(a) \vee \text{SUNK}(b) \tag{4.4}$$

These rules of interpretation have been actually realized by the function *check*( ) through the use of boolean functions, as will be shown in the following paragraphs.

It should be noticed that the actual boolean function $\mathcal{I}(w)$ associated to an expression w is identified *implicitly* by the definition of the function *check*(w), which, in some cases, is recursive.

The recursive 'call' to the function $\mathcal{I}$( ) inside the definition of *check*( ) is necessary when the *innermost* argument 'a' of the expression which is being considered can be a negative event. If this is the case, the new terms which have to be evaluated might need the reapplication of some of the simplifications 0–8.

The recursion on *check*( ) itself has been used when the truth value of the argument might be determined not simply by its presence in the current state (and thus calculated directly using the function B( )) but, instead, by the value of an associated boolean function.

For example, if w' =SBEL(x), the result of *check*(w') is defined as *check*(x) and not simply as $B(x)$. In fact, if x =SUND(E_1), for example, the truth of SUND(E_1) is determined as $\neg$ (B(E_1) $\vee$ B(NOT(E_1))), and not simply by checking if SUND(E_1) belongs to the current state.

Let us now examine, one at a time, all the cases which constitute the definition of the function *check*( ).

The cases presented take into consideration every possible expression w' :: $<WFE>$ which has been 'filtered' by the simplifications 0–8. The reader should bear in mind that the order in which the expression w' is matched against the possible cases does matter, and, indeed, should be the one given in the definition

of the function *check*( ).

The first case after the initial match of the expression w' against the tautologies and contradictions constitutes the equivalent of the rule of 'necessitation': the expression SBEL(x) is evaluated *True* iff the event x is *True* in the speaker model.

The two following cases implement a sort of 'default mechanism' for the two beliefs NOT(SR(e)) and NOT(SIEXP(s)), as explained below.

The first case assumes NOT(SR(e)) to be *True* iff the belief SR(e) is not present in the state. This definition reflects a commonly adopted behaviour, according to which if we cannot recall in our memory an event, we will assume (by *default*) that event not to be part of our past (real) experience. In other words, in case of real experiences (only), the system adopts the Closed World Assumption, in which if something is not explicitly stated, it is considered *False*.

The second one is exactly the same, only that it deals with *inductive* real experiences containing support events $s :: <Su>$.

It is interesting to notice that the definition of the two previous boolean functions forces any query in the form SUND(SR(e)) or SUND(SIEXP(s)) to be evaluated *False*. This means that the presence of these two expressions in the list of contradictions (respectively, I and II) is actually unnecessary.

The case dealing with any expression $w' :: <A>$ is reached only if the previous cases have not produced a successful match. According to this definition, all the possible expressions generated by $<A>$ (apart from the previous two) are evaluated simply by checking for their presence in the current state. In other words, these expressions are treated as elements of the inner language Li, and associated to the boolean function $B(w')$.

The following case, $w' \cong$ NOT(x), deals with any negated expression generated by $<P1>$ or $<P2>$. Notice that whilst for the expressions $e :: <E>$ it is not always true that $e \vee$ NOT(e) (for it could as well be SUND(e)), for all the other queries x of type $<P1>$ or $<P2>$ the meaning of NOT(x) *includes* the case of uncertainty, and thus it is possible to impose that $x \vee$ NOT(x) always hold. Hence, in such cases, the truth of NOT(x) has been defined simply as the logical negation of the truth value of x, that is $\neg\, check(x)$.

More formally,

$$\forall S \subseteq Li,\ \forall x :: <WFF>,\ x \not:< E>, \quad \mathcal{I}(x) \vee \mathcal{I}(\text{NOT}(x)) \mid_S = True$$

The next case implements the default mechanism for w'=SUND(HUNK(e)), the

default belief of the set $S_2(e)$, which will be evaluated *True* iff *none* of the terms present in the set $S_2(e) \setminus \{w'\}$ is evaluated *True*.

The following case evaluates any expression $w' \cong$ SUND(HUND(_)) by checking for its presence in the state (i.e. $w' \in Li$).

Case SUND(e) realizes the default mechanism for the set $S_1(e)$. Notice that because of simplification 1, the event e cannot be a negated event.

The next match, $w' \cong$ HUNK(SUP(a, b)), realizes the interpretation explained before (Equation 4.3).

Because of simplifications 7–8, the case $w' \cong$ HUNK(e_i) is reached only by expressions with $e_i :: <Ei>$. Hence, w' is simply searched for in the current state ($w' \in Li$).

The next case implements the default rule for the set $S_3(e)$, whilst the subsequent match $w' \cong$ HUND(_) — which can be reached successfully only by queries in the form HUND(e) with $e :: <E>$ — will be treated as inner language expressions (i.e. associated to $B(w')$).

The three following cases corresponds to the relations formalized at page 89, deduced from the diagram of Figure 4.2.

Query $w' \cong$ HBEL(NOT(HUNK(e))) can be thought as asking whether the hearer believes to have conceived the event 'e' in his (her) mind. The answer is 'yes' iff the event is not unknown by the hearer, i.e. $\neg$ *check*(HUNK(e)).

The next three cases are self explanatory; all the remaining expressions in the form HBEL(_) which did not match any of the previous cases are treated as elements of the inner language Li by the last case of the definition.

Through the given definition of interpretation $\mathcal{I}(\,)$, it is possible to identify the inner language $Li \subset Lo$. In fact, the inner language Li can be defined as the *union* of all the domains of the boolean functions adopted by the interpretation $\mathcal{I}(\,)$. These domains can be identified by isolating all the expressions w' of Lo which the function *check*( ) associates directly to the boolean function $B(w')$.

However, the formal definition of the inner language is the object of Section 4.2.3. The following sub-section completes the specification of the BP algorithm 'A', by listing the inference rules of the set R.

Inference Rules

The set R of inference rules adopted by the algorithm A is listed below. Each rule uses a *variable*, and represents, in fact, a *set* of inference rules. The corresponding 'grounded' set can be obtained by replacing the variable with all the possible expressions which it represents.

Two different *types* of variables have been used, each type being associated to a specific letter:

- variable e, of type $<Ei>$ ($e :: <Ei>$);
- variable s, of type $<Su>$ ($s :: <Su>$).

The production rules for $<Ei>$ and $<Su>$ are those previously defined for the syntax $<WFE>$, generating the outer language Lo.

The right-hand side (consequence) of many rules contains multiple terms, separated by commas (','). In such cases, the rule should be interpreted as representing many rules, each one having the same premiss and, as single consequence, a different term taken from the right-hand side of the considered rule.

i_1) SR(e) $\vdash e$
i_2) SIEXP(s) $\vdash s$
i_3) HR(e) $\vdash e$, HBEL(HR(e))
i_4) HIEXP(s) $\vdash s$, HBEL(HIEXP(s)), NOT(HIEXP(NOT(s)))

i_5) HBEL(SR(e)) $\vdash$ HBEL($\mathbf{C}_{R_4}${SR(e)})
i_6) HBEL(SIEXP(s)) $\vdash$ HBEL($\mathbf{C}_{R_4}${SIEXP(s)})
i_7) HBEL(HR(e)) $\vdash$ HBEL($\mathbf{C}_{R_4}${HR(e)})
i_8) HBEL(HIEXP(s)) $\vdash$ HBEL($\mathbf{C}_{R_4}${HIEXP(s)})

i_9) HBEL(SBEL(SR(e))) $\vdash$ HBEL(SBEL($\mathbf{C}_{R_4}${SR(e)}))
i_{10}) HBEL(SBEL(SIEXP(s))) $\vdash$ HBEL(SBEL($\mathbf{C}_{R_4}${SIEXP(s)}))
i_{11}) HBEL(SBEL(HR(e))) $\vdash$ HBEL(SBEL($\mathbf{C}_{R_4}${HR(e)}))
i_{12}) HBEL(SBEL(HIEXP(s))) $\vdash$ HBEL(SBEL($\mathbf{C}_{R_4}${HIEXP(s)}))

i_{13}) NOT(HR(e)) $\vdash$ HBEL(NOT(HR(e)))
i_{14}) NOT(HIEXP(s)) $\vdash$ HBEL(NOT(HIEXP(s)))

i_{15}) HBEL(SR(e)) $\vdash$ HBEL(SBEL(SR(e)))
i_{16}) HBEL(SIEXP(s)) $\vdash$ HBEL(SBEL(SIEXP(s))), HBEL(NOT(SIEXP(NOT(s))))

i_{17}) HBEL(SBEL(SR(e))) $\vdash$ HBEL(SR(e))
i_{18}) HBEL(SBEL(SIEXP(s))) $\vdash$ HBEL(SIEXP(s)), HBEL(SBEL(NOT(SIEXP(NOT(s)))))

where $R_4 = \{i_1, i_2, i_3, i_4\}$. Notice that the expressions in the form '$pred(\mathbf{C}_R\{x\})$' represent, again, a *set* of consequences, defined as follows:

$$pred(\mathbf{C}_R\{x\}) = \{pred(e) \mid e \in \mathbf{C}_R\{x\}\}$$

In words, this set contains all the expressions having predicate '*pred*' and, as argument, one of the elements of the closure $\mathbf{C}_R\{x\}$.

For example, rule i_7 can be rewritten as follows:

i_7) HBEL(HR(e)) $\vdash$ HBEL(e), HBEL(HR(e)), HBEL(HBEL(HR(e))

since $\mathbf{C}_{R_4}$\{HR(e)\} = { e, HR(e), HBEL(HR(e)) }[17]. Hence, rule i_7 is equivalent to

[17]Notice that the original element HR(e) can be removed from this closure set, as simply

the two following rules:

$$i'_7)\ \text{HBEL(HR}(e)) \ \vdash\ \text{HBEL}(e)$$
$$i''_7)\ \text{HBEL(HR}(e)) \ \vdash\ \text{HBEL(HBEL(HR}(e)))$$

Rules i_1 and i_2 are quite straightforward: if the speaker has a real (or inductive) experience, the event object of that experience will have to be believed.

Rules i_3 and i_4 correspond to rules i_1 and i_2, respectively, 'transferred' into the hearer model.

Rule i_3 should be read as follows: if the speaker believes that the event 'e' is a real experience for the hearer, then (s)he should also believe that 1) 'e' is *True*, and 2) the hearer believes 'e' to be a real experience.

In fact, if we do believe someone to have witnessed an event, then we should also believe the event to have happened. The second consequence implies that the speaker expects the hearer to be always *aware* of the experiences that (she believes) the hearer has experienced. This excludes possible states of *unconsciousness* or altered sensorial perceptions in the hearer. Moreover, this implies that the system assumes that there is no divergence of opinions with regard to real experience events which (the speaker believes) have been 'shared' by S and H.

Rule i_4 is analogous to i_3, except for the fact that one more consequence is required, for inductive experiences can contain negated events. In fact, if the hearer has (inductively) experienced the support s, then (s)he cannot possibly hold the belief in the counter example NOT(s) at the same time, as this would invalidate the whole sequence of inductive experiences[18].

The asymmetry between the consequences of i_3–i_4 and those of rules i_1–i_2 is only apparent. In fact, if we assume the premisses of rules i_1 and i_2 to hold, then we can conclude that the expressions — for the speaker model — analogous to those of rules i_3–i_4 will be evaluated *True* through the mechanism of *interpretation.*

In fact, considering i_1, it is easy to see that from the presence of the expression SR(e) in the state it follows that also the belief SBEL(SR(e)) (which is the analogous of HBEL(HR(e)) in i_3) is evaluated *True.*

As far as i_2 is concerned, from the presence of SIEXP(s) it follows immediately that SBEL(SIEXP(s)) (symmetrical of HBEL(HIEXP(s)) in i_4). Moreover, the

re-generating the premiss of the rule itself.

[18]Notice that the variable s::<Su> can also represent a negated support event NOT(SUP( )).

belief NOT(SIEXP(NOT(s))) is *True* iff SIEXP(NOT(s)) does *not* belong to the state (from the default mechanism introduced for this expression); on the other hand, SIEXP(NOT(s)) cannot coexist with SIEXP(s) (as rule i_2 would produce the consequences s and NOT(s), which are supposed to be mutually exclusive). Hence, from the presence of SIEXP(s), it follows implicitly that NOT(SIEXP(NOT(s))) (symmetrical of NOT(HIEXP(NOT(s))) in rule i_4) is *True*.

The two groups of rules i_5–i_8 and i_9–i_{12} are based on rules i_1–i_4: rules i_5–i_8 represent the equivalent of i_1–i_4 transferred into the hearer model, their premisses and consequences being essentially unchanged except for the prefix 'HBEL' which has been added in front of them; similarly, i_9–i_{12} constitute the same rules 'translated' for the hearer's model of the speaker through the addition of the predicated HBEL(SBEL( )) [19].

Rules i_{13}–i_{14} are similar to rules i_3–i_4 and are based on the same hypothesis of 'awareness' of the hearer, but draw the inference from the belief that the hearer did *not* experience a specific event: if, for some reason, we believe that the hearer did not witness an event 'e', then we should also believe that 'e' is not part of the hearer's experiences, and therefore that (s)he believes the same.

The symmetrical cases for the speaker NOT(SR(e)) $\vdash$ SBEL(NOT(SR(e))) and NOT(SIEXP(s)) $\vdash$ SBEL(NOT(SIEXP(s))) have not been included as already realized through the interpretation $\mathcal{I}$( ).

Rules i_{15}–i_{16} constitute a 'completion' of rules i_5–i_6 (they have the same premisses). Their introduction is due to the absence of the terms

SBEL(SR(e)) and
SBEL(SIEXP(s)), NOT(SIEXP(NOT(s)))

as 'explicit' consequences in rules i_1 and i_2, respectively.

The same absence would lead to the analogous completion of rules i_9–i_{10} with term HBEL(SBEL(SBEL(SR(e)))) for i_9 and HBEL(SBEL(SBEL(SIEXP(s)))), HBEL(SBEL(NOT(SIEXP(NOT(s))))) for i_{10}, which should be added to rules i_{17}–i_{18}. However, of these expressions, only the last one is generable by $<WFE>$, and has 'survived' in the consequences of rule i_{18}.

The other two consequences in rules i_{17} and i_{18} — namely, HBEL(SR(e)) and HBEL(SIEXP(s)) — derive from the consideration according to which the hearer is

[19] The adoption of the formalism of closure $\mathbf{C}_{R_4}\{\,\}$ in the definition of rules i_5–i_{12} is purely due to explanatory reasons: these rules can be easily rendered 'explicit' and rewritten as normal inference rules. Notice also that in i_{11}–i_{12} the consequences containing three levels of belief nesting (e.g. HBEL(SBEL(HBEL(HR(e))))) should not be considered, as not allowed by $<WFE>$.

likely to make the same *assumptions* about the speaker that the latter makes about the former in rules i_3–i_4, where the hearer is expected to be always aware of the real experiences (s)he has lived, excluding states of unconsciousness and uncertain memories. On the same basis, in i_{17}, the hearer (is expected to) 'deduce' that from SBEL(SR(e)) it must follow SR(e), and similarly for i_{18}, in the case of an inductive experience.

This set R of inference rules will be applied to the set of *inner* [20] beliefs $S \subseteq Li$, extending it — through the operation of *closure* — into a larger set of inner expressions $S' = \mathbf{C}_R(S)$ which will be used as current state by the BP algorithm.

It should be noticed that the set of inference rules described represents *finitely* — through the use of variables — an infinite set of inference rules: in fact, the variable s::$<Su>$ can be replaced by an infinite number of support expressions, having an arbitrarily high level of nesting; moreover, e::$<Ei>$ represents a countable infinity of 'primitive' events E_i, with $i \in \aleph$.

However, if the given state S contains a finite number of propositions, its closure $S' = \mathbf{C}_R(S)$, calculated using the given set R, is also a finite set, and the algorithm which calculates S' will always terminate. This result holds for this specific set of rules R, and is proven by the theorem reported below, which concludes this section.

Theorem 4.1 *Given a finite set S of expressions generated by $<WFE>$, the closure $S' = \mathbf{C}_R(S)$ is finite.*

Proof Each of the two arguments of a SUP( ) event is allowed to be a SUP( ) event itself, yielding to expressions with an arbitrary high level of nested supports. Since this infinite recursion is syntactically permitted only in case of a SUP( ) event, there are only two ways of obtaining an infinite set of beliefs which contain expressions of $<WFE>$: 1) to generate all the possible SUP( ) expressions; 2) to generate all the infinite number of primitive events of type $<Ei>$.

However, considering the inference rules i_1–i_{18}, none of them increases the level of support nesting or introduces new primitive events E_i, and, therefore, none of them can be used to produce an infinite set of $<WFE>$ expressions from an initial one which is finite.

□

[20] The formal definition of the inner language Li will be given in the next section.

4.2.3 The Inner language

From the definition of interpretation $\mathcal{I}(\,)$, it can be seen that the truth of any query w' of Lo has been associated with a boolean function which is either the identity $B(w')$ (in which case $w' \in Li$), or an expressions containing elements taken from the three sets S_1–S_3 of mutually exclusive and exhaustive beliefs. For example,

$$\begin{aligned}
\mathcal{I}(\text{NOT(SUND(NOT}(e)))) &= B(e) \vee B(\text{NOT}(e)) \\
\mathcal{I}(\text{HBEL(NOT(HBEL}(e)))) &= B(\text{HUND}(e)) \vee B(\text{HBEL(NOT}(e)) \\
\mathcal{I}(\text{HUND(NOT(SUNK}(e)))) &= B(\text{HUND(SUNK}(e)))
\end{aligned}$$

Moreover, the expressions of Lo which have been associated to the identity function belong *all* to the sets S_1–S_3.

Hence, the expressions in S_1–S_3 are the 'basic' semantic units which can be used to identify the truth of any other expression of Lo, and represent a good set of candidates to form the inner language $Li \subset Lo$.

As a matter of fact, the set M, defined as

$$M = \bigcup_{e::<E>} (S_1(e) \cup S_2(e) \cup S_3(e))$$

contains all of the expressions of Li plus other propositions which do not strictly need to be included in Li. This 'surplus' consists of some of the contradictory expressions (more precisely, pairs I–I' and II–II') and the three *default* beliefs, SUND(e), SUND(HUNK(e)) and HUND(SUNK(e)).

The resulting inner language $Li \leftarrow <WFF>$ (*Well Formed Formulas*), obtained by removing the mentioned expressions from M, is described by the following BNF formalization:

$$\begin{aligned}
<WFF> &::= <A> \mid <Sund>) \mid <Hunk> \mid <Hund> \mid <Hbel> \\
<Sund> &::= \text{SUND(HUND(}<Eh>\text{))} \\
<Hunk> &::= \text{HUNK(}<Ei>\text{)} \\
<Hund> &::= \text{HUND(}<Eh>\text{)} \mid \text{HUND(SUND(}<Es>\text{))} \\
<Hbel> &::= \text{HBEL(}<A>\text{)} \mid \text{HBEL(SUND(}<Es>\text{))} \mid \\
&\quad\ \text{HBEL(SBEL(}<A>\text{))} \mid \text{HBEL(SUNK(}<Ei>\text{))} \\
<Es> &::= \text{HR(}<Ei>\text{)} \mid \text{HIEXP(}<Su>\text{) |SUP(}<A>,<A>\text{)} \mid <Ei> \\
<Eh> &::= \text{SR(}<Ei>\text{)} \mid \text{SIEXP(}<Su>\text{) |SUP(}<A>,<A>\text{)} \mid <Ei>
\end{aligned}$$

where the production rules for the symbols $<A>$, $<E>$, $<Su>$ and $<Ei>$ are those already specified in $<WFE>$.

Although the expressions NOT(SR(e)) and NOT(SIEXP(e)) will not be considered members of the inner language Li, as not associated to the identity function, the symbol $<A>$ in the production rule $<WFF>::=<A>$ of this formalization has been left unchanged, for reasons of clarity.

Notice that the truth of any support query SUP(a, b) is determined through a direct check for its presence in the current state S'. This implies that every support relation which holds must be explicitly stated, but it also means that the syntax can be easily extended to include expressions such as SUP(HBEL(a), b): in fact, this can be done simply by replacing every symbol $<A>$ with $<WFF>$ in all the SUP($<A>$,$<A>$) expressions which appear in $<WFE>$ and $<WFF>$.

In order to complete the correct definition of the belief system, it is necessary to guarantee that the properties of mutual exclusivity and exhaustivity required for the (outer) expressions of the sets S_1–S_3 are always satisfied (see page 87). These conditions can be guaranteed by imposing a set of *restrictions* on the inner language Li. In fact, any restriction imposed on the inner language Li will indirectly produce a 'projected' restriction on the outer language Lo.

In other words, limiting the admissible inner states $S \subseteq Li$ of the system is reflected by a limitation of the possible 'outer' states, where the outer state $S^* \subseteq Lo$ of a system with state $S \subseteq Li$ is defined as $S^* = \{w \in Lo \mid BP(w, S) = True\}$. Hence, if appropriate restrictions are imposed on Li, no outer state S^* will ever violate the required conditions for S_1–S_3.

The specific associations adopted for the three default beliefs w_1 =SUND(e), w_2 = SUND(HUNK(e)) and w_3 = HUND(SUNK(e)) guarantee that each default will be evaluated *True* iff none of the remaining elements of the corresponding sets $S_1(e)$, $S_2(e)$ and $S_3(e)$ belongs to the (inner) state.

For example, given $e = \mathrm{E}_1$, the belief SUND(E_1) will be evaluated *True* by the system iff neither E_1 nor NOT(E_1) belongs to the current (extended) state.

Because of this specific characteristic of the interpretation $\mathcal{I}(\,)$, the restrictions imposed on Li can be simply reduced to a condition of *mutual exclusivity* on the sets $S_1 \setminus \{w_1\}$, $S_2 \setminus \{w_2\}$ and $S_3 \setminus \{w_3\}$.

In fact, for $S_1(e)$, for example, it is sufficient to impose the following condition

of mutual exclusively on Li

$$\mathcal{U}_1) \quad \neg(e \wedge \mathrm{NOT}(e))$$

to obtain the terms of $S_1(e)$ to be mutual exclusive *and exhaustive* in Lo. Notice that the imposition of the restriction $\mathcal{U}_1$ on Li corresponds to the adoption, for the belief system, of the epistemic version of axiom D of the normal modal logics (see page 25).

The other two restrictions $\mathcal{U}_2$ and $\mathcal{U}_3$ are deduced analogously from the sets S_2 and S_3.

4.3 Implementation

The belief system described in the previous section has been implemented in Gofer [78], a non-strict semantics (i.e. '*lazy*' evaluation) functional programming environment supporting a language syntactically and semantically similar to Haskell, which is a pure functional language with lazy evaluation and polymorphic-class based type checking system (see [75]).

The choice of a functional language for the realization of the system is due to the specific characteristics of the BP algorithm. First of all, its mathematical formulation, based on the definition of the recursive function of interpretation $\mathcal{I}(\,)$, and the use of the mechanism of pattern matching, are particularly suitable for a functional language implementation.

Moreover, the representation of *sets* of propositions can be realized through the *list* data structure, for which Gofer (and Haskell) provides specific built-in support. For example, the feature of 'lazy' evaluation allows the management of lists containing an infinite number of elements, as the actual value of these elements is calculated only when strictly necessary for the specific computation required by the user.

The system implemented accepts in input any query correctly generated by the syntax $<WFE>$, and evaluates its truth according to the BP algorithm 'A' and to the set of inner propositions contained in the current state $S \subset Li \leftarrow <WFF>$.

It should be pointed out that the set of tautologies (and contradictions) has been realized in the implementation through a list of expressions — containing *typed variables* — which is automatically appended to the current state. Hence, for example,

the query q=SUND(SR(E_{34})) will produce the answer (*False*, NOT(SUND(SR(_)))), indicating that q is a contradiction, as its negation NOT(q) matches the tautology NOT(SUND(SR(x))), which is *True* for any string $x ::< E >$.

As illustrated by the previous example and, more extensively, by the following one, besides the boolean value representing the truth value of the query, the system has been programmed to return also the 'reasons' which have led to the specific result.

Example 4.2 Given the state

$$S = \{\mathrm{SR}(\mathrm{E}_1), \mathrm{HBEL}(\mathrm{SR}(\mathrm{E}_1))\}$$

the query $w = \mathrm{NOT}(\mathrm{HUNK}(\mathrm{E}_1))$ will generate the answer

$$(\mathit{True}, \mathrm{HBEL}(\mathrm{E}_1))$$

which indicates that the belief w holds because of the presence of HBEL(E_1) in the completed state $S' = \mathbf{C}_R(S)$. In fact, using the inference rules in R, S' results to be

$$S' = S \cup \{\mathrm{E}_1, \mathrm{HBEL}(\mathrm{E}_1), \mathrm{HBEL}(\mathrm{SBEL}(\mathrm{E}_1)), \mathrm{HBEL}(\mathrm{SBEL}(\mathrm{SR}(\mathrm{E}_1)))\}$$

which contains HBEL(E_1). Notice that the query is evaluated *True* as

$$\mathcal{I}(w') \mid_{S'} = \neg\, B(\mathrm{HUNK}(\mathrm{E}_1)) \mid_{S'} = \neg\, \mathit{False} = \mathit{True}$$

The 'reason' justifying the result of a query is determined according to the following method: if the result of a direct match $B(w)$ is *True*, then the presence of w itself in the state is considered as the reason for the answer, and the expression w is returned. This would have happened, in the previous example, had the query w been any of the propositions in S'.

If the check $B(w)$ is *False*, the reason returned is identified through the sets of mutually exclusive and exhaustive beliefs S_1–S_3. In fact, the set containing the 'missing' expression (in the example, $S_2(\mathrm{E}_1) \ni \mathrm{HUNK}(\mathrm{E}_1)$) must contain one — and one only — 'alternative' belief which holds instead: this belief is returned as the justification of the result. (In the previous example, the belief currently holding in S_2 was HBEL(E_1)).

It should be noticed that the expression HUNK(E_1) is also a member of the set $S_3(E_1)$. Hence, this set contains a *second* reason — namely, HBEL(SBEL(E_1)) — justifying the result. The latter justification has been considered of *secondary* importance, if compared to the reason taken from $S_2(E_1)$; in fact, the set S_2 is used to represent the model of the hearer, whereas S_3 contains the expressions describing the model of the speaker in the hearer's view, a less 'direct' and normally uncertain set of information.

In the actual implementation, together with each the reason, the system returns also a *qualifier* and a list of *links* attached to it. The qualifier indicates the type of the proposition which has been returned as reason. There are four possible qualifiers: '*Lax*' (logical axiom), '*Dflt*' (default proposition), '*Act*' (active belief) and '*Cons*' (consequence derived from other beliefs). A 'logical axiom' is simply a tautology (or contradiction), that is, a belief which has been adopted as valid axiom by the system. 'Default' indicates that the belief holds as a result of the absence of other beliefs. An 'active' belief is simply one of the proposition which have been included in the core state, specified initially. A 'consequence' is a belief which has been derived through the inference rules from other beliefs (beliefs of the core state or other consequences).

The list of *links* is always empty, except when the reason is qualified as 'Cons': in this case, the links 'point' to the proposition(s) constituting the premiss(es) from which such belief has been derived. Notice that in the current implementation, if p is derived from q, and q is derived from r, the link of p will point directly to r[21].

Because of the characteristics of the definition of the function of interpretation, it is also possible that one query is evaluated *True* (or *False*) because of more than one reason. For example, the query HUNK(SUP(a, b)) is associated with the boolean expression $\mathcal{I}$(HUNK(a)) $\vee$ $\mathcal{I}$(HUNK(b)), which, in turn, will have its terms evaluated. If both of the terms are *True*, then both will constitute a valid reason, and should be returned in the result.

In order to account for such possibilities, the system has been programmed to return, for each query, a *list of lists* of reasons. The structure of list of lists allows to represent *any* possible boolean combination of reasons; in fact, any boolean expression can be always reduced to a *disjunction of conjunctions* of terms ('Or-And' form) or, equivalently, to a *conjunction of disjunctions* of terms ('And-Or'

[21] The 'links' have been realized simply by effecting a gödelisation of the propositions present in the state, so that a link is represented by a natural number which 'points' to a specific proposition.

form). Therefore, for example, the expression

$$\text{And-Or [} [a, b], [c] \text{]}$$

should be interpreted as

$$((a \vee b) \wedge (c))$$

Suppose, in the previous example, that only the first of the two terms HUNK(a), HUNK(b) is evaluated *True*, whereas the other one is found to be *False* (because, for example, HBEL(b)). In this situation, the complete expression returned by the belief prover would be as follows:

$$\text{Or-And [[(}True, Act, \text{HUNK}(a)\text{)], [(}False, Act, \text{HBEL}(b)\text{)]]}$$

The overall boolean value of the above expression is *True* because of the presence of the first sublist containing a single reason evaluated *True*. Notice that if the query had been NOT(HUNK(SUP(a, b))), i.e. the negation of the previous one, the result would have been simply deduced by negating the result of the previous query, transforming the 'Or-And' into an 'And-Or' and negating every single reason:

$$\text{And-Or [[(}False, Act, \text{HUNK}(a)\text{)], [(}True, Act, \text{HBEL}(b)\text{)]]}$$

Notice that the overall boolean value of this expression has now become *False*.

The system has been endowed with two specific functions for the evaluation of a query, namely, '*prove()*' and '*check()*'. The latter function returns the complete And-Or (or Or-And) expression representing the result, regardless of its overall value. The former actually evaluates the result of *check()*, and returns only the reason(s) (if there are any) which make the global expression *True*. If the overall value of the expression is *False*, the object '[[ ]]' is returned instead.

The complete belief system is composed by nine separate modules, four of which (Env_man.lgs, Relations.lgs, ParserKit.lgs and OurTypes.lgs) have been taken from Dr. Maria Fox and Dr. Derek Long's implementation of the management of the variable unification, parsing of expressions and types definition developed for the AbNLP planning system [45].

4.3.1 Testing

The testing of the system has been performed through a set of queries which force the execution of the code for all of the significant cases of the BP algorithm A. The

state given in input to the system is reported below, preceded by the command which has been invoked to display it:

```
? unlines (map show axioms_Set)
SR(E1)
SUP(E1,E2)
E2
HBEL(SR(E1))
HBEL(SUP(E1,E2))
HUND(E2)
HBEL(SUND(E2))
HR(E3)
HUNK(E4)
```

Part of the list of queries adopted for testing the system is reported below, followed by the corresponding results produced.

```
tst1 = prove set_T (NOT [HBEL [NOT [HBEL [E1]]]])
tst2 = check set_T (SUND [E1])
tst3 = prove set_T (NOT [SUND [E1]])

tst4 = prove set_T (SUND [E45])
tst5 = check set_T (HUNK [E1])

tst6 = prove set_T (NOT [HUNK [E1]])
tst7 = check set_T (E45)
tst8 = check set_T (NOT [E45])

tst9 = check set_T (NOT [SBEL [NOT [HUNK [SUP [E4,E3]]]]])
tst10 = check set_T (HBEL [NOT [SUNK [SUP [E4,E3]]]])
tst11 = check set_T (SUND [HUNK [SUP [E4, E3]]])
tst12 = check set_T (HUND [SUNK [SUP [E4, E3]]])

? show tst1
Or-And [[(Cons,HBEL(E1))[44]]]
? show tst2
And-Or [[(False,(Cons,E1)[41])]]
? show tst3
Or-And [[(Cons,E1)[41]]]
```

```
?  show tst4
And-Or [[(Dflt,SUND(E45))[]]]
?  show tst5
And-Or [[(False,(Cons,H_BEL(E1))[44])]]
?  show tst6
Or-And [[(Cons,HBEL(E1))[44]]]
?  show tst7
Or-And [[(False,(Dflt,SUND(E45))[])]]
?  show tst8
Or-And [[(False,(Dflt,SUND(E45))[])]]
?  show tst9
Or-And [[(True,(Act,HUNK(E4))[])],[(False,(Cons,HBEL(E3))[48])]]
?  show tst10
Or-And [[(True,(Lax,NOT(SUNK(-1)))[])]]
?  show tst11
And-Or [[(False,(Act,HUNK(E4))[])],[(True,(Cons,HBEL(E3))[48])]]
?  show tst12
Or-And [[(False,(Act,HUNK(E4))[])],[(True,(Dflt,HUND(SUNK(E3)))[])]]
```

The parameter `set_T` passed to the functions *check()* and *prove()* consists of the extended set of beliefs (set of 'Theorems') obtained as the closure of the core set `axioms_Set`. The pointers returned at the end of the 'Cons' reasons (e.g., in test no. 11, the link [48]) refer to a specific proposition of the set `set_T`, which contains also, at the beginning, the complete list of tautologies ('logical axioms').

Notice, for example in test no. 10, the result indicating the match with a tautology. In the proposition returned, the uninstantiated variable of type $< E >$ is represented by a negative number (-1).

The total execution time required by the system to evaluate any of these queries is smaller than five seconds. However, it should be pointed out that, in the environment used for running the tests, the code is executed by an *interpreter*, and that the program itself has not been optimised for speed.

Chapter 5

Integration of a Belief System with a Planning System

This chapter analyses, from an abstract point of view, the feasibility and realization of the integration between a belief system (implemented according to the theoretical paradigm described in Chapter 3) and a generic planning system.

The first section describes the basic elements which characterize the type of planners considered, defines explicitly the problem of integration and introduces the method of integration adopted.

The second section illustrates some theoretical results, definitions and examples which are useful for the following discussion.

Finally, the last section contains the detailed description of the mechanisms constituting the integration algorithm and of the formal theory upon which its correctness relies.

5.1 Introduction to the Problem

Before analysing the method adopted to integrate the beliefs system with a planner, it is necessary to define the general characteristics of a planning system, and to identify clearly what the 'integration problem' consists of.

5.1.1 The Planning System Paradigm

The following definition identifies the general concept of planning problem, and introduces the basic components of a standard planning system:

Definition 5.1 A standard *Planning Problem* can be identified as a triple (I, Op, G), where

1. I is the initial *state*, containing a collection of propositions describing a given 'state of the world';
2. Op constitutes the set of *operator schemes*, i.e. the list of possible actions that can be used to transform the initial state. Each operator schema normally contains *variables*, which are instantiated to specific values at the moment of the operator application;
3. G represents the set of *goals*, and contains all the propositions which are required to be 'True' in the *final* state.

A *solution* to a planning problem consists of a *plan*[1] which transforms the initial state into the final one, in which all the goals are achieved.

Given the initial problem, the functioning of a planning system is basically determined by the *goal-achieving procedure*, constituting the algorithm which drives the process of choosing the appropriate actions (steps) to be inserted in the plan in order to achieve a specific goal.

Many different versions of these fundamental elements have been developed during the history of planning (for a useful account of work in planning, see for example [96]). Nonetheless, there are few basic components which, although named differently by different authors, can be found in most of the planners currently in use.

First of all, instances of operators are almost universally intended to be interpreted as actions being applicable only under specific *preconditions* and producing a set of *effects* on the current state. The preconditions consist of a list of propositions which must be true when the operator is applied.

Secondly, the goal-achieving procedure normally includes the possibility of the 'step addition' mechanism, a process consisting of finding an operator which contains in its *effects* the specific goal to be achieved, and adding it to the plan.

[1] A plan can be thought of as an ordered sequence $< O_1, O_2, \ldots, O_n >$ of possibly repeated *instances* of operator schemes, called *steps*.

One of the first and most influential planners, STRIPS [38], introduced the idea of representing the effects of an operator with two lists of propositions, called 'ADD' and 'DELETE' lists, containing respectively the list of predicates to be added to and deleted from the state at the moment of the operator's application. The analysis of the integration of a BP algorithm into a generic planning system will assume the planning problem to be defined with this kind of operators, containing (at least) *preconditions, add* and *delete* lists and representable as triples (P,A,D). Notice that the presence of the P-A-D lists in an operator constitutes a necessary condition, but should not be seen as an upper limit to the complexity of the formalism. In other words, such lists are only taken to be the 'minimum' level of expressiveness required by the operator schemes in order to be suitable for the process of integration. As it will be pointed out in Chapter 7 (end of Section 7.3.1), the integration process implemented can be applied directly to more sophisticated operators, or easily modified to allow the use of a more expressive formalism (see Section 5.3.1), including, for example, operators with conditional effects.

5.1.2 The Integration Problem

In order to identify clearly the aim of the integration process, let us recall the main features which a belief system, implemented according to the theoretical model explained in Chapter 3, presents.

A belief system is formally defined as a triple (Lo, Li, A), where Lo is the outer language, $Li \subseteq Lo$ is the inner language, and A is a Belief-Prover (BP) algorithm for Lo on Li.

One of the characteristics of the BP algorithm consists of allowing the definition of complex belief-situations using only a relatively small set of unequivocal inner expressions, still enabling the system to be queried through the wider set of complex expressions taken from the outer language Lo. In other words, the BP simplifies the process of complete definition of a belief state by reducing to Li the redundant language Lo, but maintains unchanged the power of expressiveness.

Therefore, the integration of a BP in a planning system should allow the definition of the initial state I of the plan through the simple and concise inner language, while permitting the use of more complex outer language expressions in the *preconditions* of the operators. In fact, a list of preconditions can simply be considered as a list of *queries* given to a belief system which adopts the current planning state as the current *belief* state.

These considerations lead to the following definition of 'BP-Planning Problem':

Definition 5.2 Given a BP algorithm for Lo on Li, a *BP-Planning Problem* is a tuple $(I, Op, G, BP(\,))$, where $I \subseteq Li$ is the initial state, $G \subseteq Lo$ is the set of goals, Op is the set of operators in the form (P,A,D) (with P$\subseteq Lo$, A$\subseteq Li$, D$\subseteq Li$), and $BP(\,)$ is the function calculated by the BP algorithm.

Notice that both preconditions and *goals* are allowed to contain any *outer* expressions, whereas the add and delete lists A and D need only to specify the essential *inner* beliefs to be added and removed from the current state $S \subseteq Li$.

In order to build a correct *plan* which solves a given BP-planning problem, a planning system should use the $BP(\,)$ function as an 'interface' between outer and inner beliefs, so that any solution found satisfies the following definition:

Definition 5.3 Given a BP-planning problem $\mathcal{P} = (I, Op, G, BP(\,))$, a *Plan* for $\mathcal{P}$ consists of a sequence of steps $< O_0, O_1, \ldots, O_n >$ (with $O_i \in Op$) such that, if S_i is the state before the application of the operator O_i and F is the final state[2], then the two following conditions are true:

1. for each step O_i =(P,A,D) in the plan,

$$\forall p \in \mathrm{P}, \quad BP(p, S_i) = True$$

2. $\forall g \in G, \quad BP(g, F) = True$

Therefore, given a specific BP-Planning Problem $\mathcal{P} = (I, Op, G, BP(\,))$ and a planning system able to solve a generic planning problem as described in the previous section, the problem of the *integration* of the BP with the given planner consists of producing a *new* system which generates correct plans to solve $\mathcal{P}$.

Example 5.1 Consider the BP algorithm 'A' described in the previous chapter, with set of inference rules R and interpretation $\mathcal{I}(\,)$ defined through the simplifications 0–8 and the function *check*( ).

Assume that all the operators of the set Op are represented by a single *operator*

[2]Notice that $S_0 = I$.

schema defined as $O_1(a, a') = (\text{P,A,D})$, where:

$$\begin{aligned} P &= \{\,\text{SBEL}(a),\ \text{SBEL}(\text{SUP}(a, a')),\ \text{SUND}(a')\,\} \\ A &= \{\,a'\,\} \\ D &= \emptyset \end{aligned}$$

and $a, a' :: <A>$.

This operator represents a simple version of the *Modus Ponens* rule: if the speaker believes a, and also believes that $\text{SUP}(a, a')$, then (s)he should conclude a'. However, the condition $\text{SUND}(a')$ must be verified in order for O_1 to be applicable.

Notice that since an operator produces a *change* in the current state, its definition must take into account the specific *restrictions* imposed on the inner language (as a result of the restrictions imposed on the outer) of the belief system, which must be satisfied by any state considered during the planning process.

For example, consider the event a': it belongs to the set of mutually exclusive and exhaustive expressions $S_1(a') = \{\ a',\ \text{NOT}(a'),\ \text{SUND}(a')\ \}$. Hence, when $\text{SUND}(a')$ is *True*, the current state cannot contain either of the two events a' or $\text{NOT}(a')$.

Through the operator O_1, the opinion of $S_1(a')$ which is currently held by the speaker changes from $\text{SUND}(a')$ into a' (notice that the addition of a' to the state implies that the belief $\text{SBEL}(a')$ will be evaluated *True*). However, even though the event a' does belong to the add list A, the proposition $\text{SUND}(a')$, unexpectedly, does not appear in the delete list D (as a matter of fact, D is empty).

Formally speaking, this is due to the fact that the proposition $\text{SUND}(a')$ is not *allowed* to appear in D, as it is not a member of the *inner* language $Li \leftarrow <WFF>$ (by definition of BP-planning problem, the lists A and D can contain exclusively inner expressions). Nevertheless, the actual reason for this 'absence' has to be found in the fact that the opinion SUND( ) has been chosen as the *default* belief of the set S_1. In fact, this means that the explicit presence in the state of the proposition $\text{SUND}(a')$ is avoided by assuming it to be *True* whenever neither of the other two beliefs a', $\text{NOT}(a')$ are present in the state (see definition of the function *check*(SUND( ))).

Therefore, the addition of the belief a' to the state will *automatically* cause the belief $\text{SUND}(a')$ to be evaluated *False*. On the other hand, since the default belief itself was not present in the state, it does not have to be deleted.

Let us now assume the initial state I to be

$$I = S_0 = \{\, \mathrm{SR}(\mathrm{E}_1),\ \mathrm{SIEXP}(\mathrm{SUP}(\mathrm{E}_1, \mathrm{E}_2)) \,\}$$

and the goal set G to contain a single proposition:

$$G = \{\, \mathrm{NOT}(\mathrm{SUND}(\mathrm{NOT}(\mathrm{E}_2))) \,\}$$

Notice that the goal constitutes an outer expression of *Lo* which does *not* belong to the inner language *Li*. A possible real interpretation of the above events could be the following:

E_1 = "John threw the stone into the river."
E_2 = "The stone sank in the water."

Within this interpretation, the goal (for the speaker) would consist of reaching a conclusion on the validity of the event $\mathrm{NOT}(\mathrm{E}_2)$ = "The stone did not sink in the water."

Given this BP-planning problem, representable as $\mathcal{P} = (I, Op, G, BP(\,)\,)$, and a 'standard' planner, the integration of the BP algorithm should produce a new planning system able to find the plan $<O_1(\mathrm{E}_1, \mathrm{E}_2)>$ which, indeed, solves the problem $\mathcal{P}$. In fact, given the initial state I, the preconditions of O_1 — with $a = \mathrm{E}_1$ and $a' = \mathrm{E}_2$ — follow immediately from the application of the inference rules i_1 and i_2. Moreover, the addition of the belief E_2 to the state will cause the goal $\mathrm{NOT}(\mathrm{SUND}(\mathrm{NOT}(\mathrm{E}_2)))$ to be evaluated *True*, according to the definition of interpretation $\mathcal{I}(\,)$.

The previous example underlines an important point: the restrictions (mutual exclusivity of the expressions of S_1–S_3) imposed on the inner language *Li* must be respected by the initial state I and by any other state obtained by applying the operators of the set *Op*. This requires the set of operators made available to the planner to satisfy the property of *coherence*, defined below:

Definition 5.4 Given a set of restrictions on a language L, an operator O is *Coherent* iff, for any coherent state $S \subseteq L$ to which O can be applied, the new state S' obtained from the application of O to S is still coherent.

The definition of coherent state has been given in Section 3.2.

A second issue which the previous example obviously gives rise to concerns the method that the new planning system will have to adopt in order to find a plan solution for the given BP-problem. The analysis of two possible approaches which can be followed to solve this problem is the object of the following section.

5.1.3 Possible Approaches to the Integration

The definition of correct *plan* for a BP-planning problem suggests that the most direct and intuitive way of integrating the model of beliefs into a planner consists of using the BP *inside* the planner as a sort of 'interface' between outer and inner languages, so that every outer-language query concerning the current state will be 'filtered' and handled by the BP.

However, this approach presents many difficulties, which arise when trying to realize an actual implementation. For example, most of the planners currently in use allow to represent the set of operators as a collection of *schemes*, enabling preconditions, add and delete lists to contain variables of a specified 'type', like $O_1(a, a')$ in the previous example.

Although this representation is employed as a way of synthesizing a larger set of possible operators, the variables are normally considered as components of the operators themselves even during the plan-construction phase. In fact, the step addition process is normally realized by *matching* a specific goal 'g' against the propositions which appear in the add list of an operator: the result of the match, if positive, will supply the value(s) which can be substituted to the variable(s) so that the operator considered achieve the goal g.

However, in case of a BP-planning problem, a simple match between a goal and the elements of the add list of an operator does not always produce a correct result.

In fact, reconsider the example introduced in the previous section: the match of the goal g = NOT(SUND(NOT(E_2))) against the (only) term of the add list A of O_1, namely, $a' :: <A>$, would not return any admissible instantiation of the variable a', as the goal g does not belong to the set of expressions of type $<A>$ ($g :: <P1>$). Nevertheless, if a' =E_1, the operator O_1 does achieve the goal g. Unfortunately, this substitution is only evident if we take into account the *internal functioning* of the BP algorithm, which evaluates *True* the goal g if the current state contains the belief E_2.

This problem is due to the fact that while the add and delete lists contain only inner expressions, the goals (and preconditions) are allowed to contain more complex outer-language propositions, which are achieved 'indirectly', i.e. as a result of the presence or absence of other beliefs. Indeed, the ability of detecting whether a specific operator 'asserts' or 'denies' a proposition p — that is, imposes p to be evaluated, respectively, *True* or *False* — without using the content of the present state requires a complex analysis, which produces a boolean condition based on both the add and the delete list (for a more detailed discussion on this issue, see [50]).

Apart from this kind of practical obstacle, a 'direct' integration of the BP presents a fundamental disadvantage: the use of the BP algorithm as a module inside the planner requires the *modification* of the *structure* and *code* of the planner, which should be properly tailored in order to 'embody' the belief system.

The big drawback of this lies in the fact that the resulting system would constitute the integration of a *specific* belief model with a *specific* planning system, having predetermined characteristics: the integration of a different BP — or even of the same one — into a different planner would require the whole process of modification to be repeated, with the consequent misuse of time and resources which would be necessary to understand and modify a completely new piece of software.

The alternative approach, which has been adopted in the actual implementation of a planning system for BP-planning problems, is based on the considerations that follow.

The adoption of the BP algorithm during the planning process is due to the fact that while the state is defined using the inner language Li, the preconditions of the operators and the goals can also be outer-language expressions. This requires the use of the BP as an 'interface' between state (and A/D lists) and preconditions (and goals) whenever the truth of an outer belief needs to be assessed.

However, if also the *state* were allowed to contain expressions of Lo, together with the add and delete lists of the operators, then the *whole* process of planning could be carried out in a outer-language 'world', with no need for any use of a belief prover.

In other words, if the state S were 'extended' with *all* of the outer beliefs which are evaluated *True* by $BP(\cdot, S)$, and the A/D lists of each operator were completed with all the *outer* propositions which are asserted and denied by the operator, the BP-problem would be *transformed* into an equivalent *standard* planning problem,

entirely defined on the language *Lo*.

Example 5.2 Let us recall the example of the previous section, in which the initial state I contained only two beliefs:

$$I = S_0 = \{\, \mathrm{SR}(\mathrm{E}_1),\ \mathrm{SIEXP}(\mathrm{SUP}(\mathrm{E}_1, \mathrm{E}_2)) \,\}$$

The operator scheme $O_1(a, a') = (\mathrm{P,A,D})$ was defined as

$$\begin{aligned} P &= \{\, \mathrm{SBEL}(a),\ \mathrm{SBEL}(\mathrm{SUP}(a, a')),\ \mathrm{SUND}(a') \,\} \\ A &= \{\, a' \,\} \\ D &= \emptyset \end{aligned}$$

with $a, a' :: <A>$, and the only goal g was $\mathrm{NOT}(\mathrm{SUND}(\mathrm{NOT}(\mathrm{E}_2)))$.

If we extend the initial state I with all the beliefs of *Lo* evaluated *True* by the BP, we obtain the set I^*, containing the closure $\mathbf{C}_R(I)$ (which is still a subset of *Li*) and an *infinite* set of outer expressions:

$$\begin{aligned} I^* &= \{\, \mathrm{SR}(\mathrm{E}_1),\ \mathrm{E}_1,\ \mathrm{SIEXP}(\mathrm{SUP}(\mathrm{E}_1, \mathrm{E}_2)),\ \mathrm{SUP}(\mathrm{E}_1, \mathrm{E}_2) \,\} \\ &\cup \{\, \mathrm{SBEL}(\mathrm{E}_1),\ \mathrm{NOT}(\mathrm{SBEL}(\mathrm{E}_2)),\ \mathrm{NOT}(\mathrm{SBEL}(\mathrm{E}_3)), \ldots, \\ &\quad \mathrm{SBEL}(\mathrm{SUP}(\mathrm{E}_1, \mathrm{E}_2)),\ \mathrm{NOT}(\mathrm{SBEL}(\mathrm{SUP}(\mathrm{E}_3, \mathrm{E}_1))), \ldots, \\ &\quad \mathrm{SUND}(\mathrm{E}_3),\ \mathrm{SUND}(\mathrm{E}_4),\ \mathrm{SUND}(\mathrm{E}_5), \ldots \} \end{aligned}$$

The extension of the operator O_1 will give $O_1^* = (P^*, A^*, D^*)$, where

$$\begin{aligned} P^* &= P = \{\, \mathrm{SBEL}(a),\ \mathrm{SBEL}(\mathrm{SUP}(a, a')),\ \mathrm{SUND}(a') \,\} \\ A^* &= \{\, a', \mathrm{SBEL}(a'), \mathrm{NOT}(\mathrm{SUND}(a')), \mathrm{NOT}(\mathrm{SUND}(\mathrm{NOT}(a'))), \\ &\quad \mathrm{NOT}(\mathrm{SBEL}(\mathrm{SUND}(a'))), \mathrm{NOT}(\mathrm{SBEL}(\mathrm{SUND}(\mathrm{NOT}(a')))), \mathrm{SBEL}(\mathrm{SBEL}(a')),\ \} \\ D^* &= \{\ \mathrm{NOT}(\mathrm{SBEL}(a')), \mathrm{SUND}(a'), \mathrm{SUND}(\mathrm{NOT}(a')), \\ &\quad \mathrm{SBEL}(\mathrm{SUND}(a')), \mathrm{SBEL}(\mathrm{SUND}(\mathrm{NOT}(a'))), \mathrm{NOT}(\mathrm{SBEL}(\mathrm{SBEL}(a'))) \,\} \end{aligned}$$

From this new 'explicit' standard planning problem, it appears clear that A^* will contain the goal g when $a' = \mathrm{E}_2$, and that the list P^* of preconditions is immediately satisfied by the initial state I^* if $a = \mathrm{E}_1$.

Notice that $P^* = P$. In fact, even if the preconditions contained only inner expressions, it would not be necessary to extend them, as they do not *modify* the state. Moreover, it should be noticed that A^* and D^* are finite. This is because the completed lists do not have to contain *all* the outer beliefs which result *True* (or *False*) after the application of O_1, but have simply to perform the necessary *changes* (additions and deletions) to maintain the coherence of the state, that is, to respect the *restrictions* imposed on *Lo*. For example, the belief SBEL(SBEL(a)) will actually be *True* after the application of the operator, but it does not have to be added to the state, as it was already present *before*. In fact, one of the preconditions of O_1 is SBEL(a'); if the extended state S^* — to which the operator is applied — contains *all and only* the outer expressions evaluated *True* by the function $BP(\cdot, S)$, then the presence of SBEL(a) in S^* must have been caused by the presence of a in S. Hence, also the belief SBEL(SBEL(a)) must have been evaluated *True*, and, thus, included in the state S^*.

It should be pointed out that the actual implementation of this method will not require the generation of the *complete* list of expressions contained in the state I^*: by adopting a 'lazy' approach, only the beliefs which are strictly needed by the discourse planning process will need to be included, reducing to a finite, small set the initial state actually calculated. The criterion used for the selection of the relevant beliefs and the motivations which justify its adoption will be explained in detail in Section 6.3.

Summarizing, the indirect approach to the problem of integration, introduced in this section, is based on the idea of *transforming* the initial BP-problem into an equivalent *standard* planning problem through the application of an 'integrator' system. Such system will *preprocess* the given initial state, goals and operators by 'extending' them with the appropriate *outer* expressions, derivable through the application of the specific belief prover algorithm (i.e., of the $BP(\,)$ function).

The same kind of approach has been also adopted by Gazen and Knoblock in [56], as mentioned earlier (p. 53). In their work, Gazen and Knoblock present a preprocessor for converting UCPOP-planning problems into Graphplan problems (see Section 2.4). The UCPOP notation allows, amongst other things, the use of 'axioms' (the equivalent of inference rules) and of operators with conditional effects. A method for the transformation of conditional operators into an 'equivalent' set of STRIPS-style (non-conditional) operators, alternative to the combinatorial

algorithm proposed by Gazen and Knoblock, has been developed by Anderson and Weld [7], and will be described in Section 5.2.2.

The inference rules (or 'axioms', in this context) are treated differently: they are not used to 'extend' the various elements of the original planning problem, as in the approach described above, but, instead, are converted directly into 'deduce' operators, having preconditions and postconditions equivalent, respectively, to the premisses and the consequences of the rule. Although this approach might seem quite simple and promising, it presents various drawbacks, as pointed out by Garagnani in [53]. For example, adopting this method of conversion, since any 'deduced' proposition p may lose its validity after each step, the preprocessing phase needs also to identify the operators that modify the propositions from which p can be derived, and to add an effect which *negates* p to each of these operators. This forces the 'axiom' (inference rule) to be re-applied in a latter step if the deduced proposition p is needed for another operator. As Gazen and Knoblock point out themselves, in the worst case, an axiom may need to be re-asserted after each step [56, p.224]. Also, as a consequence of these 'corrections', the operators produced are likely to be conceptually incorrect, and negate propositions which, according to the 'meaning' of the considered operator, should not be negated (cf. the 'put-on' operator of their example [56, p.225]). Finally, the planning state obtained by applying the steps specified by a correct plan solution does not necessarily end up containing all the consequences which can be deduced at that point using the inference rules.

Because of these negative aspects, the alternative method for preprocessing of the inference rules introduced earlier, based on their application (through the use of the $BP()$ function) for the 'extension' of the initial state and of the add and delete lists of the operators, has been adopted instead.

Finally, it should be noticed that the indirect (or preprocessing) approach to the problem of the integration offers, in general, two important advantages. First of all, the new (standard) problem resulting from the pre-extension process can be solved directly by a large number of planners, without requiring *any modifications* of the structure and code of the planning system specifically chosen for the task.

Secondly, the process of transformation of the initial problem, which substitutes the actual integration of the BP algorithm, can be performed *automatically* and for a complete *class* of belief systems built according to the paradigm described in Chapter 3.

The following sections contain the detailed description of this approach, along with the formal theory upon which it relies.

5.2 Indirect Integration

The indirect approach, introduced in the previous section, is based on the following consideration: given a set of inner belief S, instead of using, whenever necessary, the BP algorithm to evaluate the truth of any outer (non-inner) belief, it should be possible to render *explicit* the results of the BP, and to *pre-generate* the set $S^* \subseteq Lo$ containing all and only the expressions of Lo evaluated true by $BP(\cdot, S)$.

This suggests that, instead of integrating the belief system into a planner by building a new planning system which makes use of the BP algorithm, it should be possible to render fully *explicit* the specific BP-planning problem $\mathcal{P} = (I, Op, G, BP(\,))$ given, by *completing* it with all the expressions of Lo which would be evaluated *True* by the BP. This new equivalent[3] problem $\mathcal{P}' = (I', Op', G')$ would not require the use of a $BP(\,)$ 'interface' function, and could be solved by a normal planner as a standard planning problem which uses expressions of Lo.

This process can be realized if the BP algorithm is transformed into an 'extension' procedure which, once supplied with the set of inner propositions $S \subseteq Li$, will generate the set $S^* \subseteq Lo$ containing S plus all of the outer propositions which would have been evaluated *True* by the function $BP(\cdot, S)$[4].

The first part of this section concerns the theoretical and practical feasibility of this transformation.

5.2.1 Composed Closure

Given a specific BP algorithm, each 'inner' state $S \subseteq Li$ will determine a specific set $S^* = \{\omega \in Lo \mid BP(\omega, S) = True\}$, containing all (and only) the outer expressions evaluated $True$ by the system. Therefore, the BP algorithm constitutes a *filter* through which it is possible to see an 'outer' state $S^* \subseteq Lo$, identified by the 'inner' S. More formally, the function $BP(\cdot, S)$ calculated by a BP algorithm for a given state $S \subseteq Li$ constitutes a boolean evaluation of Lo. In fact, since $BP(\omega, S)$ returns a boolean value for every $\omega \in Lo$, $BP(\cdot, S)$ can be considered as a function

[3] A formal definition of *equivalence* will be given subsequently.

[4] It is interesting to point out that this transformation can be seen as turning the 'backward-reasoning' part of the BP — i.e. the syntactical transformations and boolean functions association — into a 'forward-reasoning' mechanism, consisting only of *inference rules*.

assigning each element of Lo to one element of the set { *True, False*}, that is, as a boolean evaluation of Lo.

Definition 5.5 Given a BP algorithm for Lo on Li, the *Outer Extension* of a state $S \subseteq Li$ consists of the set $S^* = \{\omega \in Lo \mid BP(\omega, S) = True\}$. The *function* which transforms each S into its outer extension S^* will be indicated as $BP^*(\,)$.

By definition of 'state identified' by a boolean evaluation (see equation 3.1), it follows that $S^* = BP^*(S)$ is the state *identified* by the boolean evaluation $BP(\cdot, S)$.

The task performed by the BP algorithm is, in a sense, summarized by the set produced by the corresponding outer extension function $BP^*(\,)$: the function $BP^*(S)$ renders explicit and immediately available *all* the possible results which would have been calculated by the BP algorithm for a specific state S.

Intuitively, in order to complete the given problem $\mathcal{P}$ it will be necessary to have every 'implicit' element of $\mathcal{P}$ *pre-extended* by the function $BP^*(\,)$, so that the global effects of the BP algorithm will be 'pre-compiled' into the initial problem, leaving the original planning system *unchanged.*

More precisely, the new initial state $I' = BP^*(I)$ will consist of the original set I plus all the expressions of Lo which would have been evaluated *True* by the $BP(\,)$ function, whereas every operator will have its effects extended so to add (or delete) the *new* outer propositions which the BP would have evaluated *True* (or *False*) after the application of the operator itself.

The next theorem shows how, given any BP, it is possible to identify the corresponding outer extension function $BP^*(\,)$ through an *algorithmic* procedure. This procedure constitutes the core of the proof of the theorem.

Before presenting the theorem, though, it is useful to introduce the concept of 'acyclic' inference rules, which will be used to describe part of the results of the proof.

Definition 5.6 A set R of inference rules is *Acyclic* iff $\forall r \in R$, the consequence 'c' of rule 'r' never appears in any of the premisses of the rules in R.

Basically, a set of acyclic inference rules is such that the consequence of the application of one of the rules can never 'activate' (nor 'forbid') the application of

another rule of the set.

In other words, if R is a set of acyclic rules on L, the closure $\mathbf{C}_R(S)$ of a state $S \subseteq L$ can be calculated simply by adding to S the set of consequences K which can be obtained *directly* from S: $\mathbf{C}_R(S) = S \cup K$, where

$$K = \{\, c \in L \mid c \notin S, \exists r \in R : r = (\mathcal{P} \vdash c),\ \mathcal{P} \mid_S = True \,\}$$

Theorem 5.1 (Composed Closure) *Given a BP algorithm which uses a set of inference rules Ri on a language Li and calculates an interpretation $\mathcal{I}()$ of Lo on Li, let $BP^* : 2^{Li} \rightarrow 2^{Lo}$ be the corresponding outer extension function, that is $BP^*(S) = \{\omega \in Lo \mid BP(\omega, S) = True\}$. There exists a set Ro of (acyclic) inference rules on Lo*

$$\begin{array}{lcl} \mathcal{P}_1 & \vdash & \omega_1 \\ \mathcal{P}_2 & \vdash & \omega_2 \\ \vdots & \vdots & \vdots \\ \mathcal{P}_n & \vdash & \omega_n \end{array}$$

such that

$$\forall S \subseteq Li, \quad BP^*(S) = \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S)) \tag{5.1}$$

Proof $\forall \omega \in (Lo \setminus Li)$, the boolean function $\mathcal{I}(\omega)$ associated to ω can be rewritten as a *disjunction* of *conjunctions*:

$$\mathcal{I}(\omega) = P_1 \vee P_2 \vee \ldots \vee P_h$$

where each conjunction P_i contains the k_i terms

$$P_i = (\, t_{i1} \wedge t_{i2} \wedge \ldots \wedge t_{ik_i} \,)$$

and each term t_{ij} can be either $B(x)$ or $\neg B(x)$, with $x \in Li$.

Consider the set Ro of inference rules on Lo built from an initial empty set by

adding, for *each* expression $\omega \in (Lo \setminus Li)$, the following list of inference rules:

$$\begin{array}{lll} P_1 & \vdash & \omega \\ P_2 & \vdash & \omega \\ \vdots & \vdots & \vdots \\ P_h & \vdash & \omega \end{array}$$

First of all, since $\omega \in Lo \setminus Li$ and all of the premisses are functions of expressions of Li, the resulting Ro constitutes a set of *acyclic* inference rules on Lo.

Moreover, Ro is such that $\forall S \subseteq Li, \quad BP^*(S) = \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

To prove this, let us see first that $BP^*(S) \subseteq \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

Let $S \subseteq Li$ be the current state, and let $\omega \in BP^*(S)$. By definition, we have

$$\omega \in BP^*(S) \Leftrightarrow BP(\omega, S) = True$$

According to the definition, $BP(\omega, S) = True$ in two possible cases:

i) $\omega \in (Lo \setminus Li) \ \wedge \ \mathcal{I}(\omega) \mid_{S'} = True$

ii) $\omega \in Li \ \wedge \ \omega \in S'$

with $S' = \mathbf{C}_{Ri}(S)$. In the first case, since $\omega \in (Lo \setminus Li)$, then the boolean function associated to ω is

$$\mathcal{I}(\omega) = P_1 \vee \ldots \vee P_h = \mathcal{F}_\omega$$

But $BP(\omega, S) = True$; hence, $\mathcal{F}_\omega \mid_{S'} = True$, which means that one of the terms $P_1, \ldots P_h$ must have been evaluated *True* in the state $S' = \mathbf{C}_{Ri}(S)$. Since these terms are *all* premisses of inference rules in Ro, then the rule whose premiss is *True* in $\mathbf{C}_{Ri}(S)$ will cause the addition of ω to the closure $\mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

In the second case, since the closure of a set always contains the original set (see equation 3.4), then $\omega \in \mathbf{C}_{Ri}(S) \ \Rightarrow \ \omega \in \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

Therefore, in both of the cases *i)* and *ii)*, $\omega \in \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$, and thus $BP^*(S) \subseteq \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

Let us now prove that $\mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S)) \subseteq BP^*(S)$.

Let $S \subseteq Li$ be the current state, and $c \in \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$. Once again, the proof can be divided into two cases:

a) $c \in (\, \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S)\,) \setminus \mathbf{C}_{Ri}(S))$

b) $c \in \mathbf{C}_{Ri}(S)$

In the first case, 'c' belongs to the 'outer' closure — obtained applying the rules in Ro — but not to the 'inner' closure $\mathbf{C}_{Ri}(S)$, obtained applying the rules in Ri. Therefore, c must be one of the consequences of the rules in Ro, i.e. there exists a rule $r \in Ro$

$$r)\ \ \mathcal{P} \vdash c$$

which has been applied to produce the final outer closure $\mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$.

Because of the way in which the set Ro has been defined, the premiss $\mathcal{P}$ contains only *inner* expressions, whilst the consequences of the rules in Ro cannot be inner expressions, as they belong to the set $Lo \setminus Li$. Therefore, the premiss $\mathcal{P}$ cannot have been made *True* by *any* of the rules in Ro, and must have been already holding in $\mathbf{C}_{Ri}(S)$.

On the other hand, the presence of the rule $r)$ in Ro must have been originated by the interpretation of 'c'

$$\mathcal{I}(c) = P_1 \vee \ldots \vee \mathcal{P} \vee \ldots \vee P_h$$

which contains the term $\mathcal{P}$. But since $\mathcal{P}$ was *True* in $S' = \mathbf{C}_{Ri}(S)$ (and $c \in Lo \setminus Li$), then $\mathcal{I}(c) \mid_{S'} = True$: therefore, $BP(c, S) = True$, hence $c \in BP^*(S)$.

In case *b)*, $c \in \mathbf{C}_{Ri}(S) = S'$ implies that $c \in Li$, as Ri contains only inference rules on Li, and $S \subseteq Li$. From the definition of $BP(\,)$, it follows directly that $BP(c, S) = True$; consequently, $c \in BP^*(S)$.

In conclusion, in both of the cases *a)* and *b)*, it results $c \in BP^*(S)$; hence, $\mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S)) \subseteq BP^*(S)$.

Since the implicitly was proved in the first part, and no assumptions were ever made regarding the current state S, then we can conclude that

$$\forall S \subseteq Li, \quad BP^*(S) = \mathbf{C}_{Ro}(\mathbf{C}_{Ri}(S))$$

□

The previous theorem shows that the 'global' output of a BP algorithm can be calculated as the closure $\mathbf{C}_{Ro}(S')$ of the extended state $S' = \mathbf{C}_{Ri}(S)$ using a set of acyclic inference rules Ro.

In other words, once identified the set of rules Ro, the BP algorithm can be *transformed* into the calculation of a composition of closures.

Indeed, it should be noticed that the proof describes a general method to deduce the set Ro from any given interpretation $\mathcal{I}(\,)$, and that the computational load of this procedure is *proportional* to the number of 'associations' required to define the function $\mathcal{I}(\,)$ completely.

This number can be assumed to be smaller than a finite constant k: in fact, even when the domain of the function $\mathcal{I}(\,)$ (namely, $Lo \setminus Li$) is an *infinite* set, it is possible to represent finitely an infinite number of associations through the adoption of *variables* (the following example actually presents this specific situation).

Finally, it is important to underline that the premisses of the 'new' set of rules Ro correspond to the conjunctions extracted from the boolean functions adopted by the interpretation $\mathcal{I}(\,)$. Hence, if these boolean functions present specific characteristics or regularities, the analogous qualities will be transferred into the inference rules of Ro.

Example 5.3 Let us recall the example introduced in Section 3.2. The outer language Lo was defined as

$$Lo = \{\, m(s),\ f(t),\ married(u,v),\ wife(w,x),\ husband(y,z) \,\}$$

whilst the inner $Li \subset Lo$ contained

$$Li = \{\, m(s),\ f(t),\ married(u,v),\ \}$$

Suppose the interpretation $\mathcal{I}(\,)$ consists of two associations:

$$\begin{aligned} \mathcal{I}(\omega_1) = &\quad \mathcal{I}(wife(x,y)) &&= married(x,y) \wedge f(x) = P_1 \\ \mathcal{I}(\omega_2) = &\quad \mathcal{I}(husband(x,y)) &&= married(x,y) \wedge m(x) = Q_1 \end{aligned}$$

Let the set $Ri = \{r\}$ contain the only inference rule on Li

$$r)\ \ married(x,y) \vdash married(y,x)$$

Notice that the domain $Lo \setminus Li$ of $\mathcal{I}(\,)$ consists of the set $\{husband(x,y),\ wife(w,z)\}$,

which is *infinite* if the set of possible names is infinite.

Applying the method described in the theorem, Ro will contain the two rules

$$r_{o1}) \ P_1 \ \vdash \ \omega_1$$
$$r_{o2}) \ Q_1 \ \vdash \ \omega_2$$

or, more explicitly,

$$r_{o1}) \ married(x,y) \wedge f(x) \ \vdash \ wife(x,y)$$
$$r_{o2}) \ married(x,y) \wedge m(x) \ \vdash \ husband(x,y)$$

Notice that these two rules together with $r)$ constitute the same set of inference rules considered in the original example. If the current state is $S = \{married(Paul, Mary), m$ the extended state will consist of the inner closure

$$S' = \mathbf{C}_{Ri}(S) = S \cup \{married(Mary, Paul)\}$$

whereas the final outer closure will be

$$\mathbf{C}_{Ro}(S') = S' \cup \{husband(Paul, Mary), wife(Mary, Paul)\}$$

It is not difficult to see that this set contains, indeed, all and only the expressions of Lo which would be evaluated *True* by the BP algorithm, that is, the set $S^* = BP^*(S)$.

By looking at the previous example, the implementation of an automatic procedure which performs the transformation of $\mathcal{I}(\,)$ into the corresponding rules constituting Ro would seem straightforward. However, this is true only if the function $\mathcal{I}(\,)$ is defined as a finite list of explicit associations between propositions of $Lo \setminus Li$ and boolean functions of inner propositions.

However, the interpretation function could be available only in an *implicit* form, i.e. $\mathcal{I}(\,)$ could be defined *recursively*, or through the use of other functions (as in the case of the belief system described in Chapter 4).

In such situation, a *general* automatic procedure that '*deduces*' the explicit associations of each expression $w \in Lo \setminus Li$ from the definition of $\mathcal{I}(\,)$ and represent them *finitely* (i.e. using appropriate variables) is actually unfeasible if the domain $Lo \setminus Li$ is *infinite*.

This implies that, in such cases, the realization of this part of the process will have to be done 'by hand'.

However, even in those cases, the method described in the theorem of Composed Closure can be used as a 'datum point'. Moreover, the theorem still guarantees the *existence* of a set of rules Ro satisfying the requirements; the problem which is left to the user consists of *finding* these rules, and representing them finitely.

The next subsection will introduce the formal definition of equivalent planning problems, constituting the basis for the development of the theory of indirect integration, which is based on the repeated transformation of a BP-planning problem into an equivalent one.

5.2.2 Equivalent Planning Problems

According to the definition of outer extension $BP^*(S)$ of a state S,

$$\forall p \in Lo,\ \forall S \subseteq Li, \quad BP(p,S) = True \Longleftrightarrow p \in BP^*(S)$$

Therefore, the definition of correct plan to solve a BP-planning problem, given in Section 5.1.2, is equivalent to the following one, in which all the expressions using the $BP(\,)$ function have been replaced with the corresponding forms which use the $BP^*(\,)$ function:

Definition 5.7 Given a BP-planning problem $\mathcal{P} = (I, Op, G, BP^*(\,))$, a *plan* for $\mathcal{P}$ consists of a sequence of steps $<O_0, O_1, \ldots, O_n>$ (with $O_i \in Op$) such that, if S_i is the state before the application of the operator O_i and F is the final state, then the two following conditions are true:

1. for each step O_i in the plan, if P is the preconditions list of O_i, then

$$\forall p \in \mathrm{P},\ \ p \in BP^*(S_i) \tag{5.2}$$

2.

$$\forall g \in G,\ \ g \in BP^*(F) \tag{5.3}$$

Notice that a BP-planning problem identified by the tuple $(I, Op, G, BP(\,))$ can now be described using the function $BP^*(\,)$ instead of the original $BP(\,)$, since *all*

of the conditions for its solution have been rewritten using the function $BP^*()$. Moreover, because of its definition, a BP-planning problem becomes a 'normal' planning problem if $BP^* = I$ (the identity function) and $Lo = Li$.

Definition 5.8 Given two planning problems $\mathcal{P}'$, $\mathcal{P}''$, if there exists a *1–1 onto* relation between the plans which solve $\mathcal{P}'$ and the plans which solve $\mathcal{P}''$, then the problems $\mathcal{P}'$ and $\mathcal{P}''$ are *equivalent.*

In other words, in order for $\mathcal{P}'$, $\mathcal{P}''$ to be considered equivalent, there must be a *bijective* function which associates every plan solution of $\mathcal{P}'$ with one (and only one) solution of $\mathcal{P}''$, and implicitly.

5.3 Transforming a BP-planning Problem

This section contains the description of two algorithms, A1 and A2, which constitute the basis of the process of indirect integration of a BP algorithm into a planner.

The integration itself consists of transforming a BP-planning problem $\mathcal{P} = (I, Op, G, BP^*)$ into an equivalent problem $\mathcal{P}' = (I', Op', G', BP_1^*)$ having $BP_1^* = Id$ [5]. If the function BP_1^* calculates the identity, in fact, its use in the problem is not necessary: from the definition of plan solution of a BP-planning problem, it follows immediately that $\mathcal{P}' = (I', Op', G', Id)$ can be solved normally, as a standard planning problem.

As previously mentioned, the transformation is realized by rendering fully *explicit* the BP-planning problem $\mathcal{P}$, that is, by completing every state and every operator with all the expressions of the *outer* language which would have been 'visible' only through the BP algorithm. These expressions are generated by the outer extension function BP^*, which can be deduced from the BP (theorem of Composed Closure).

The resulting problem will be entirely defined in an outer-language 'world', and will not require the use of a belief prover for the construction of the plan solution.

Let us now suppose that the function BP^* can be *decomposed* into many sepa-

[5] Id represents the 'identity' function, i.e. $Id(S) = S$, $\forall S \subseteq Li$.

rate functions:

$$BP^* = f_n \circ f_{n-1} \circ \ldots \circ f_2 \circ f_1$$

where each component calculates a *closure* using a specific set of rules.

If that were the case, instead of integrating the whole function BP^* into the problem $\mathcal{P}$ in one single step, the transformation could be split into many separate sub-integrations: the first one will integrate the function f_1, producing an *equivalent* problem with a new outer extension function

$$BP_1^* = f_n \circ f_{n-1} \circ \ldots \circ f_2$$

The second will integrate f_2, producing a new problem with

$$BP_2^* = f_n \circ f_{n-1} \circ \ldots \circ f_3$$

until, after n similar steps, the last integration of f_n will lead to the final BP-planning problem with

$$BP_n^* = Id$$

which will be solved by a standard planning system.

Indeed, because of the theorem of Composed Closure, the function BP^* has already been shown to be composed of two separate closures:

$$BP^* = \mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}$$

Moreover, if a set of rules R contains separate subsets of rules $R_1, R_2, \ldots, R_k$ such that for each $i \in \{1, \ldots, k\}$ the consequences of each subset R_i *do not appear in any the premisses of sets* $R_1, \ldots, R_{i-1}$, then it is possible to split the calculation of $\mathbf{C}_R$ into the calculation of k separated 'sub-closures'. Notice that the need for decomposing the set R of rules into blocks which satisfy the above mentioned property arises, in practice, form the presence in R of different 'types' of rules, for which different algorithms of integration have to be used.

For example, suppose the set Ro to be composed of two sets Ro_1 and Ro_2 such that

$$Ro_1 \cap Ro_2 = \emptyset, \quad Ro_1 \cup Ro_2 = Ro$$

and such that the consequences of the rules in Ro_2 never appear in any of the premisses in Ro_1.

If this property holds, the calculation of the closure $\mathbf{C}_{Ro}$ can be divided into

two independent 'blocks', and realized as

$$\mathbf{C}_{Ro} = \mathbf{C}_{Ro_2} \circ \mathbf{C}_{Ro_1}$$

as none of the rules of the second group (which are used *after* the rules of the first group) can generate propositions which would have been relevant for the calculation of $\mathbf{C}_{Ro_1}$.

Since the Theorem of Composed Closure guarantees that the set Ro, derived from the interpretation function $\mathcal{I}()$ of the BP algorithm, is *acyclic*, then the decomposition of Ro into blocks will always be possible, and *any possible grouping* of the rules is equally correct.

By splitting Ro into two sets Ro_b and Ro_a, the function BP^* can be rewritten as

$$BP^* = \mathbf{C}_{Ro_b} \circ \mathbf{C}_{Ro_a} \circ \mathbf{C}_{Ri}$$

and the complete integration can be divided into three separated steps, each realizing the sub-integration of one of the three functions, that is, effecting the integration of a different set of inference rules.

The algorithm A1, explained below, has been designed to be used for the first and third steps of the integration, i.e., for the rule sets Ri and Ro_b, which result to contain rules of the same type, i.e., in the form $p \vdash c$.

5.3.1 A1

Let $\mathcal{P} = (I, Op, G, BP^*)$ be a BP-planning problem such that $BP^* = f_o \circ f_i$, with $f_i = \mathbf{C}_R$, $f_o = \mathbf{C}_{Ro}$, and R,Ro two sets of inference rules on Lo.

If:

1. R is *acyclic*[6];
2. R contains only inference rules in the form $p \vdash c$, with $p, c \in Lo$;

then the algorithm A1, consisting of the application of the two procedures Ω_1 and Γ_1, transforms $\mathcal{P}$ into an equivalent BP-planning problem $\mathcal{P}' = (I', Op', G', BP_1^*)$, where:

[6]The definition of acyclic set of rules has been given at page 131.

- $I' = \Omega_1(I)$
- $Op' = \Gamma_1(Op)$
- $G' = G$
- $BP_1^* = f_o$

The formal proof of the correctness of this transformation will be given after the complete description of the algorithm, which includes the definitions of $\Omega_1()$ and $\Gamma_1()$.

Intuitively, the transformation A1 integrates into the planning problem itself the *effects* of the function $f_i = \mathbf{C}_R$: as a result, the function $BP^* = f_o \circ f_i$ is reduced to $BP_1^* = f_o$ in the new problem $\mathcal{P}'$.

In this new problem, each state will explicitly contain all the propositions which would have been previously added by f_i, i.e. by the closure $\mathbf{C}_R$. It follows that the new 'extended' initial state I' should contain all the propositions which would have been generated by $\mathbf{C}_R(I)$: hence, $\mathbf{C}_R(I) \subseteq I'$.

This gives a first idea of the form of the transformation $\Omega_1()$, producing I' from I. However, let us firstly analyse the function $\Gamma_1()$, whose specifications will also lead to the complete definition of Ω_1.

$\Gamma_1()$ transforms each operator $O \in Op$ into a new operator $O' = \gamma_1(O)$ which may contain *conditional effects*. As we shall see, the presence of conditional effects is due to the use of the inference rules.

Let O =(P,A,D) be the operator which is being considered. The new operator O' will consist of a list of triples $[(P', A', D'), e_1, e_2, \ldots, e_n]$, where the head (P', A', D') represents the main effect of the operator, and the following e_i are conditional effects $e_i = (P_i, A_i, D_i)$.

The new precondition list P' will simply be equivalent to the original P.

Consider now the add list A. The new operator O' must add all the propositions which would have been added by O. Hence, $A \subseteq A'$. Moreover, O' should also add all the 'new' propositions which the closure $\mathbf{C}_R(A \cup S)$ would have contained after the addition of A to the state S.

For example, suppose that $p \in A$, and that one of the inference rules of R consists of $p \vdash c$: the consequence 'c' should appear in the new add list A', for

it would have belonged to $\mathbf{C}_R(A) \subseteq \mathbf{C}_R(A \cup S)$ [7]. So should *all* the possible consequences of the set A obtained using the inference rules of R; in other words, A' will have to contain the set $\mathbf{C}_R(A)$.

Consider now the new delete list D'. The operator O deletes all the propositions in D; should O' delete the set D, too? Suppose that an element $d \in D$ is such that the set R contains a rule $p \vdash d$. If the proposition p is still present in the state after the application of O, the closure $\mathbf{C}_R$ of this state will contain, again, the proposition d, as the rule $p \vdash d$ will generate a *new* element d from p.

Therefore, the elements of D will have to be deleted by O' under a *specific condition*, that is, that the considered proposition d be not the *consequence* of a rule $p \vdash d$ such that p is present in the current state (or will be present in the state after the application of the operator O).

In order to formalise this condition, let us define the set of 'premisses' of a proposition c as

$$Prem(c) = \{p_1, \ldots p_k\} \tag{5.4}$$

where the terms $p_1, \ldots p_k$ constitute the premisses of *all* the inference rules in R having consequence 'c' and premiss $p_i \notin D$ (if $p_i \in D$, the premiss p_i will not belong to the state after the application of O).

The 'negation' of this expression, written as

$$\neg\, Prem(c)$$

will represent the set of negated propositions $\{\neg p_1, \neg p_2, \ldots, \neg p_k\}$, corresponding, when interpreted as a conjunction, to the boolean expression

$$\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_k$$

According to the previous considerations, each proposition $d \in D$ (which *does not belong to* A') will produce the addition of a conditional *deletion*

$$e_i = (P_i, A_i, D_i) = (\neg Prem(d), \emptyset, \{d\})$$

to the list (initially empty) of conditional effects $[e_1, \ldots e_n]$ in O' [8].

[7] Notice that, because of the specific form of the rules in R, assumed by point 2. of the hypotheses, it is not difficult to prove that $\forall A, B \subseteq L$, $\mathbf{C}_R(A \cup B) = \mathbf{C}_R(A) \cup \mathbf{C}_R(B)$.

[8] Notice that the use of the condition $\neg Prem(d)$ as precondition list requires the negated premisses $\neg p_1, \ldots, \neg p_k$ to be expressed each by a ***single*** expression of Lo. This requirement is satisfied by the language $Lo \leftarrow <WFE>$ (see page 85).

In other words, d will be deleted by O' iff it would have not appeared in the closure $\mathbf{C}_R(S \cup A \setminus D)$, where S is the state to which the operator O is applied.

The new unconditional delete list D' can be left empty. However, in case $Prem(d) = \emptyset$, the current state cannot contain any premiss of rules which could have prevented the deletion of d. Hence, the condition $\neg Prem(d)$, with $Prem(d) = \emptyset$, should be evaluated *True*, and the resulting effect will consist of an *unconditional* deletion, which can be merged (at the very end) with the main deletion D'.

Consider now the list $L_D = [e_1, \ldots, e_m]$ of newly added conditional effects, with $e_i = (\neg Prem(d), \emptyset, \{d\})$. Suppose a proposition d constitutes the premiss of one of the inference rules of R, e.g. $d \vdash c$. If the condition $\neg Prem(d)$ is verified, and d is deleted, should c also be deleted?

The answer is 'yes' *iff* its presence in the state was caused solely by the presence of d. Indeed, there might be another proposition q in the current state such that $q \vdash c$, in which case c should not be deleted; in alternative, 'c' could have been present in the state since the beginning of the plan (i.e., $c \in I$): in this case, c should not be deleted, either.

Given d such that $(\neg Prem(d), \emptyset, \{d\}) = e_i \in L_D$ and

$$r)\ \ d \vdash c$$

where r is one of the inference rules in R, there are two possible cases: either $c \in Li$, or $c \notin Li$.

First, consider the case of $c \notin Li$. In this case, c cannot have been one of the propositions asserted in the initial state, which is a subset of Li; neither can it have been added by the main effect of one of the previous operators, for every add list A must be a subset of Li, too.

Therefore, if c is in the state, it must be a consequence of one (or many) of the inference rules. Hence, if $c \notin A'$, the following conditional deletion should be appended to the list of effects of O':

$$e_j = (P_j, A_j, D_j) = (\neg Prem(d) \cup \neg Prem(c), \emptyset, \{c\}) \qquad (5.5)$$

Notice that the presence of the condition $\neg Prem(d)$ is actually superfluous. In fact, because of the acyclicity of R, the set $Prem(d)$ is always empty if d constitutes the

premiss of a rule $d \vdash c$ of R [9].

This process must be repeated for each d which appears as one of the delete lists of L_D. Notice, though, that because of the hypothesis of *acyclicity* of R, it is not necessary to *iterate* the process and reconsider the deletions produced by this *second* list L'_D of new conditional effects.

This follows from the fact that all the propositions c conditionally deleted by this new list are consequences of rules, and cannot appear as premisses of other rules of R. Hence, their (possible) deletion can not produce any further propagatory effect.

Consider now the second possibility, in which c (consequence of a rule r containing d as a premiss) does belong to Li.

In this case, c could have been present in the state since the initial situation, or even added afterwards by one of the operators. In order to check whether c was added to the state because of the presence of a rule r, it is necessary to distinguish, somehow, between inner propositions added 'directly' (i.e. asserted in the initial state or added by the main effect A of an operator) and those 'derived' from others already present[10].

In order to effect this distinction without keeping a track of the 'origin' of each of the propositions present in the state, it is necessary to have automatically *marked* every *inner* expression p which is derived through the inference rules.

This marking can be realized by adding to the state a proposition-marker CONS(p) every time a new proposition p, obtained using an inference rule, is added to the current state[11].

A marker CONS(p) must be introduced iff the inner proposition p, deduced as a consequence of some rule, was *not already present* in the state (and must be removed only when the proposition p is removed)[12].

The necessity of marking each newly introduced inner expressions which is derived from the inference rules of R leads to the complete definition of the extension Ω_1, used to calculate the initial state $I' = \Omega_1(I)$ of the new problem. In fact, it has

[9]The introduction of $\neg Prem(d)$ was originally due to the fact that this effect is 'deduced' as a consequence of the possible application of a conditional deletion $e_i \in L_D, e_i = (\neg Prem(d), \emptyset, \{d\})$.

[10]Notice that this distinction is necessary only for inner expressions, for the outer expressions are *always* derived from inference rules.

[11]The arbitrarily chosen predicate 'CONS' must not belong to the language Lo.

[12]The addition of these markers can be considered as an 'extension' of the language of the new BP-planning problem.

been shown that I' must contain $\mathbf{C}_R(I)$; hence, each element $j \in (\mathbf{C}_R(I) \setminus I)$ — representing a consequence of I — will require the addition of a marker CONS(j) to I'.

More formally, given a set $S \subseteq Lo$, if we define

$$\text{CONS\_}S = \{\text{CONS}(p) \mid p \in S\}$$

the transformation $\Omega_1(\,)$ can be specified as follows:

$$\Omega_1(S) = \mathbf{C}_R(S) \cup \text{CONS\_}(\mathbf{C}_R(S) \setminus S) \tag{5.6}$$

The same reasoning would seem to be applicable to the add list A' of the new operator O': for each element $a \in (\mathbf{C}_R(A) \setminus A)$, the expression CONS($a$) should be added to A'. However, this would not always be correct.

In fact, even if a proposition 'c' is a consequence of A, it could have been already present in the current state S as (unconditional) addition of a previous operator, or even carried on since the initial state I. Hence, the addition of the marker CONS(c), in these cases, would be inappropriate, as, by definition, CONS(c) must be added iff the proposition c was *not already present* in the current state.

The solution to this problem consists of using a *conditional addition* instead of adding the marker directly to A'. In other words, for each element $a \in (\mathbf{C}_R(A) \setminus A)$, a new conditional effect

$$e_i = (P_i, A_i, D_i) = (\{\neg a\},\ \{\text{CONS}(a)\},\ \emptyset)$$

must be added to the list of conditional effects of O', whereas the new add list A' will simply be calculated as $\mathbf{C}_R(A)$.

Finally, since a marker must be removed only when its argument is removed from the state, for every expression d which appears in a (conditional) delete list D_i, if $d \in Li$, then the proposition-marker CONS(d) must be appended to the list D_i.

Having fully introduced the presence of markers in the language of the new BP-planning problem, it is now possible to use them to express the required condition for the conditional *deletions*.

Each effect $e_i = (\neg Prem(c_i), \emptyset, \{c_i\}) \in L'_D$, appended to the operator O' as a consequence of a conditional deletion (that is, $e_i \notin L_D$) and such that $c_i \in Li$, will

become

$$e_i = (\neg Prem(c_i) \cup \text{CONS\_}\{c_i\},\ \emptyset,\ \{c_i,\ \text{CONS}(c_i)\})$$

In fact, this guarantees that the proposition c_i is deleted iff 1) none of the rules which can generate it will be applicable to the state, and 2) the presence of c_i in the state is due only to an inference rule $d \vdash c_i$ whose premiss is either being removed or does not belong to the current state.

Summarizing the results of the previous considerations, the transformations Ω_1 and Γ_1 of algorithm A1 can be defined as follows:

- $\Omega_1(I) = \mathbf{C}_R(I) \cup \text{CONS\_}(\mathbf{C}_R(I) \setminus I)$;
- $\Gamma_1(Op) = Op'$, such that $\forall O' \in Op',\ O' = \gamma_1(O)$, and $\gamma_1(P, A, D) = [(P', A', D'), e_1, \ldots, e_n]$, where
  - $P' = P$;
  - $A' = \mathbf{C}_R(A)$;
  - $D' = \emptyset$;
  - the list of conditional effects $[e_1, \ldots, e_n]$ is composed of three separate lists, L_A, L_D and L'_D, obtained in the following way:
    1. $\forall a \in (A' \setminus A)$, append to the list L_A the conditional addition

       $$(\{\neg a\},\ \{\text{CONS}(a)\},\ \emptyset)$$

    2. $\forall d \in D$, if $d \notin A'$, append to the list L_D the conditional deletion

       $$(\neg Prem(d),\ \emptyset,\ \{d,\ \text{CONS}(d)\})$$

    3. $\forall e_i \in L_D$, consider the conditionally deleted proposition d: $\forall c \in (\mathbf{C}_R(\{d\}) \setminus \{d\})$, if $c \notin A'$, then
       (a) if $c \notin Li$, then append to L'_D the conditional deletion

       $$(\neg Prem(c),\ \emptyset,\ \{c\})$$

       (b) if $c \in Li$, then append to L'_D the conditional deletion

       $$(\neg Prem(c) \cup \text{CONS\_}\{c\},\ \emptyset,\ \{c, \text{CONS}(c)\})$$

Before analysing the correctness of this algorithm, it should be pointed out that its applicability relies upon two main assumptions concerning the characteristics

of the inference rules of R. First of all, the left hand side of each rule is allowed to contain only *one* term; secondly, the set of rules R must be *acyclic.*

Although with these assumptions the algorithm A2 has been formally proved to be correct, their relaxation does not necessarily imply that the BP-planning problem cannot be integrated.

In fact, even if one of these hypotheses does not hold, it is still possible to specify a *different* and more sophisticated algorithm which performs the integration, but the formal correctness of which will be harder to prove and will rely upon other hypotheses.

For example, if the set R were not acyclic, points 2 and 3 of the algorithm, calculating the conditional deletion lists L_D and L'_D, would constitute the first two cycles of a *loop*, in which every iteration executed would possibly produce a new list of conditional effects derived from the previous one. The condition for the termination of this loop would rely on the specific characteristics of the rules, and, more precisely, on the *finiteness* of the procedure calculating the set $\mathbf{C}_R$. An algorithm which implemented such characteristics was not found to be obviously defective on testing.

Another important aspect of the algorithm A1 consists of the fact that it can easily extended to pre-process operators with conditional effects: given an operator containing a *list* of triples $e_i = (P_i, A_i, D_i)$ (where $e_0 = (P, A, D)$ constitutes the main precondition and effects), A1 can simply be repeatedly applied to each triple of the list, and its output — lists of conditional effects — appended to the operator.

Moreover, it should be noticed that the requirement of acyclicity of the set R can be replaced by a weaker condition, requiring only that R be able to be divided into an ordered sequence of (acyclic) *blocks* of rules, such that the premisses of the rules of each block do not appear as premisses in any of the rules of the preceding blocks (as described at page 139).

Finally, in [53], Garagnani proposes an extended algorithm to perform the integration of an acyclic set of rules with left hand sides containing boolean expressions with *many* terms, in the form $p_1 \wedge p_2 \wedge \ldots \wedge p_k \vdash c$. Such algorithm consists essentially of a variation of A1, and allows the use of operators with conditional effects.

Before embarking ourselves in the formal proof of the correctness of the algorithm A1, it will be useful to examine the results which it produces when applied to a practical example.

Example 5.4 Consider the 'translation' into the hearer-model domain of the *Modus Ponens* operator adopted in Examples 5.1 and 5.2:

Persuade(a, b)
P = { HBEL(a), HBEL(SUP(a, b)), HUND(b) }
A = { HBEL(b) }
D = { HUND(b) }

This operator represents a simplified version of the 'Persuade(a, b)' operator scheme which will be adopted in Chapter 6 for the implementation of a real discourse planning system (see Section 6.3.1). Consider also the two following operators:

Spoil(a, b)
P = { HBEL(a), HBEL(SUP(b, a)), NOT(HBEL(b)) }
A = { HUND(a) }
D = { HBEL(a) }

Assert-Real(a)
P = { SR(a), HUNK(SR(a)) }
A = { HBEL(SR(a)) }
D = $\emptyset$

These three operator schemes represent the basic strategies of persuasion upon which the discourse planning system implemented in Chapter 6 will rely. Their characteristics and functioning will be explained in details in Section 6.3.1; for the moment, and for the aims of this example, it will be sufficient to give only an intuitive description of their meaning.

The intention which lies behind 'Spoil(a, b)' consists of leading the hearer to become uncertain about a belief 'a' — relying upon specific supporting evidence 'b' — by pointing out the uncertainty of the support belief 'b' itself[13].

The operator 'Assert-Real(a)', instead, represents a 'primitive' speech act, in which the speaker simply asserts that the event 'a' constitutes a real experience and, in virtue of the Sincerity Axioms, expects the hearer to believe such assertion[14].

Let us now examine the effects that the transformation Γ_1 has upon these operators, assuming the set of inference rules R to contain the rules $\{i_1, i_3, i_5, i_7, i_{13}\}$,

[13]The condition NOT(HBEL(b)) could be already holding in the initial situation, orelse be achieved as a subgoal.

[14]As mentioned before, the model exclude the possibility of agents being unreliable about their own real experiences.

taken from the list i_1–i_{18} of rules for the belief system described in Chapter 4 (see page 107); these five rules have actually been adopted as set of rules for the system implemented in Chapter 6.

After the application of A1, the new 'Persuade' operator will be identical to the original, except for the addition of the marker CONS(HUND(b)) to the delete list D due to step 2 of the algorithm:

γ_1**(Persuade**(a, b)**)**
P = { HBEL(a), HBEL(SUP(a, b)), HUND(b) }
A = { HBEL(b) }
D = { HUND(b), CONS(HUND(b)) }

Notice that $Prem$(HUND(b)) = $\emptyset$, hence the conditional deletion has been 'inglobated' into the main effects of the schema. Similarly, A1 adds a conditional deletion to Spoil(a, b), which, in this case, cannot be inglobated in the operators' main effects:

γ_1**(Spoil**(a, b)**)**
P = { HBEL(a), HBEL(SUP(b, a)), NOT(HBEL(b)) }
A = { HUND(a) }
D = $\emptyset$
P_1 = { NOT(HBEL(SR(a))), NOT(HBEL(HR(a))) }
A_1 = $\emptyset$
D_1 = { HBEL(a), CONS(HBEL(a)) }

The lists of conditional effects which A1 built in order to produce the new operator are $L_A = []$, $L_D = [e_1 = (P_1, A_1, D_1)]$ and $L'_D = []$. The conditional effect e_1 in L_D, generated by the step 2, presents, as condition, the result of the expression $\neg Prem$(HBEL(a)); In other words, its two terms constitute the negation of the premisses of the two inference rules i_5 and i_7, whose consequences match with HBEL(a)[15].

Finally, 'Assert-Real(a)' is extended into the operator scheme reported below:

[15]Notice that this transformation has been simplified by assuming $a ::< Ei >$.

γ_1(**Assert-Real**(a))
P = { SR(a), HUNK(SR(a)) }
A = { HBEL(SR(a)), HBEL(a) }
D = $\emptyset$
P_1 = { NOT(HBEL(a)) }
A_1 = { CONS(HBEL(a)) }
D_1 = $\emptyset$

In this case, the three lists of conditional effects were $L_A = [e_1]$, $L_D = [\,]$ and $L'_D = [\,]$. The add list A of the main effect (e_0) has been extended through rule i_5, and the marker CONS(HBEL(a)) will be added iff HBEL(a) is not already holding at the moment of the operator application.

In conclusion, after the subsequent transformation of the given initial state I into $I' = \Omega_1(I)$ (as defined by A1), the inference rules of R will have been completely integrated into the planning problem, allowing the system to *ignore* them during the whole process of planning: the new set of operators and initial state can simply be adopted by any standard planner to achieve the assigned set of goals.

It should be noticed that although this transformation can be applied to operator schemes which contain variables (as illustrated by the previous example), practical considerations have led to actually implementing the integrator system so that it effects, initially, the *instantiation* of all the variables contained in the operator schemes to produce a set of grounded operator instances on which the transformation algorithms will then be applied. Such considerations will be explained in detail in Section 6.3.

5.3.2 Correctness of A1

The proof of the correctness of A1 is based on various properties concerning the transformations Ω_1 and Γ_1.

First of all, from the definition of Ω_1

$$\forall S \subseteq Li,\quad \Omega_1(S) = \mathbf{C}_R(S) \cup \text{CONS\_}(\mathbf{C}_R(S) \setminus S)$$

it follows immediately that

$$\forall S \subseteq Li, \qquad \mathbf{C}_R(S) \subseteq \Omega_1(S) \tag{5.7}$$

and, since the marker 'CONS' does not to belong to the language Lo,

$$\forall S \subseteq Li, \quad \Omega_1(S) \cap Lo = \mathbf{C}_R(S) \tag{5.8}$$

These two propositions formalise the character of Ω_1, which might add more expressions than the normal operation of closure calculated using the set of rules R, but it is such that the elements in excess *do not belong* to the language Lo.

The fact that the new initial state I' is calculated as $\Omega_1(I)$ indicates that these new expressions, not belonging to Lo, are allowed to be used in the state descriptions of the new problem $\mathcal{P}'$, and, therefore, will have to be included in the *new* inner language Li'.

If we identify the set Δ of new expressions which can be used in the new problem as

$$\Delta = \Omega_1(Li) \setminus Li \tag{5.9}$$

(notice that $\Omega_1(Li) \supseteq \mathbf{C}_R(Li) \supseteq Li$), then the new inner and outer languages for the new problem $\mathcal{P}'$ will be, respectively:

$$Li' = \Omega_1(Li) = Li \cup \Delta \tag{5.10}$$

$$Lo' = Lo \cup \Delta \tag{5.11}$$

with $\Delta \cap Lo = \emptyset$.

It should be noticed that the outer extension function f_o of the new BP-planning problem $\mathcal{P}'$ can be considered as defined on a wider domain than the original 2^{Lo}.

In fact, since f_o calculates the *closure* of a subset of Lo, any expression given in input to f_o which is not an element of Lo is simply reported in output, as it cannot generate any application of inference rules (which have, as premisses, boolean functions of Lo). This allows f_o to accept, as arguments, sets of expressions containing also elements not belonging to Lo.

More formally, the function f_o, considered defined on the 'extended' domains $f_o : 2^{Lo'} \rightarrow 2^{Lo'}$, satisfies the following property:

$$\forall S \subseteq Lo, \ \forall D \subseteq \Delta, \qquad f_o(S \cup D) = f_o(S) \cup D \tag{5.12}$$

The second and most important property concerning the transformation functions involves both Ω_1 and Γ_1.

Consider, in the original planning problem $\mathcal{P}$, the change produced by an operator O applied to the initial state I

$$I \xrightarrow{O} S_1$$

indicating that the application of O to I produces the new state S_1.

The new problem $\mathcal{P}'$ starts with an 'extended' initial state $I' = \Omega_1(I)$, and adopts a new set of (extended) operators $Op' = \Gamma_1(Op)$, where Γ_1 is realized by transforming each operator $O = (P, A, D) \in Op$ into a new conditional-effect operator $O' = \gamma_1(O)$ [16].

Because of the way in which the transformation of the operators has been defined, the corresponding change in the new problem-world

$$\Omega_1(I) \xrightarrow{\gamma_1(O)} S'_1$$

produces a state S'_1 which is *exactly* $\Omega_1(S_1)$.

In other words, every step

$$S_t \xrightarrow{O} S_{t+1}$$

of a plan in the original problem has a corresponding step

$$\Omega_1(S_t) \xrightarrow{\gamma_1(O)} \Omega_1(S_{t+1})$$

in the new problem plan, and vice versa. More formally, given two states S_t and $\Omega_1(S_t)$, and two operators $O \in Op$ and $O' = \gamma_1(O) \in \Gamma_1(Op)$, the application of O to S_t and of O' to $\Omega_1(S_t)$

$$\begin{array}{ccc} S_t & \xrightarrow{O} & S_{t+1} \\ \downarrow & & \\ \Omega_1(S_t) & \xrightarrow{\gamma_1(O)} & S'_{t+1} \end{array} \tag{5.13}$$

produces two states S_{t+1} and S'_{t+1} such that $S'_{t+1} = \Omega_1(S_{t+1})$.

Because of this property, since the plan of the new problem $\mathcal{P}'$ starts from an initial state $I' = \Omega_1(I)$, then every following state $\Omega_1(S_t)$ of the plan in $\mathcal{P}'$ (including the final state) will correspond to a state S_t of a plan in $\mathcal{P}$.

Therefore, the two BP-planning problems $\mathcal{P}$ and $\mathcal{P}'$ can be shown to be equiv-

[16] Notice that γ_1 is *1–1*.

alent. The formal proof of this equivalence is the object of the following theorem:

Theorem 5.2 (A1-equivalence) *Let $\mathcal{P} = (I, Op, G, BP^*)$ be a BP-planning problem, with $BP^* = f_o \circ f_i$, where:*

- *f_o calculates a* closure, *i.e., there exists a set R_o of inference rules on Lo such that $f_o = \mathbf{C}_{R_o}$;*
- *$f_i = \mathbf{C}_R$, and the set R of inference rules on Lo is* acyclic *and contains only rules in the form $p \vdash c$, with $p, c \in Lo$.*

Then, the new BP-planning problem $\mathcal{P}' = (\ \Omega_1(I),\ \Gamma(Op),\ G,\ f_o\ )$ [17] produced by A1 is equivalent *to $\mathcal{P}$.*

Proof Consider a sequence of steps $Z{=}<O_1, O_2, \ldots, O_n>$ with $O_i \in Op$, and associate it with the sequence $Z'{=}<\gamma_1(O_1), \ldots, \gamma_1(O_n)>$ of operators $\in \Gamma(Op)$. This association represents a *bijection*, as the transformation γ_1 is *1–1*, and each operator in $\Gamma(Op)$ has been generated by one (and only one) operator in Op.

We need to prove that if Z solves $\mathcal{P}$, then Z' solves $\mathcal{P}'$, and vice versa.

Let

$$I \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \ldots \xrightarrow{O_n} S_n = F$$

be the sequence of states obtained applying the plan Z, and

$$\Omega_1(I) \xrightarrow{\gamma_1(O_1)} S'_1 \xrightarrow{\gamma_1(O_2)} \ldots \xrightarrow{\gamma_1(O_n)} S'_n = F'$$

the sequence generated by Z'.

Because of the property 5.13, since $\Omega_1(I) \xrightarrow{\gamma_1(O_1)} S'_1$, then $S'_1 = \Omega_1(S_1)$. Analogously, $S'_2 = \Omega_1(S_2)$, $S'_3 = \Omega_1(S_3)$, etc.

Let us see that if $S_n = F$ is a *final* state for $\mathcal{P}$ (i.e. it contains the goal set G) then $S'_n = F' = \Omega_1(S_n)$ is a final state for $\mathcal{P}'$, and vice versa.

F is a final state for $\mathcal{P}$ *iff*

$$\forall g \in G,\ \ g \in BP^*(F) \tag{5.14}$$

by definition of correct plan solution of a BP-planning problem (see equation 5.3). This is equivalent to

$$G \subseteq f_o(\ \mathbf{C}_R(F)\ )$$

[17] Notice that $\mathcal{P}'$ contains operators with conditional effects.

But since $G' = G$, this condition is the same as

$$G' \subseteq f_o(\ \mathbf{C}_R(F)\ )$$

Both members of the above relation are $\subseteq Lo$, as $\mathbf{C}_R(F) \subseteq Lo$, and f_o calculates a closure using inference rules on Lo. Thus, for any set $D \subseteq \Delta$ — which implies $D \cap Lo = \emptyset$ — the previous relation holds *iff*

$$G' \subseteq f_o(\ \mathbf{C}_R(F)\ ) \cup D$$

Because of property 5.12, this corresponds to

$$G' \subseteq f_o(\ \mathbf{C}_R(F) \cup D\ )$$

But the set D can be chosen so that $D = (\ \Omega_1(F) \setminus \mathbf{C}_R(F)\ )$; hence, the previous expression becomes

$$G' \subseteq f_o(\Omega_1(F)) = f_o(F') \tag{5.15}$$

This condition expresses the fact that F' is a final state for the new problem $\mathcal{P}'$. Since it results that 5.14 holds *iff* 5.15 holds, then we have proved that if $S_n = F$ is a *final* state for $\mathcal{P}$ then $S'_n = F'$ is a final state for $\mathcal{P}'$, and vice versa.

The second part of the proof concerns the other condition (see equation 5.2) required by the definition of plan solution for a BP-planning problem. This condition requires that for every operator O_i used in a solution, if P is its precondition list, then

$$P \subseteq BP^*(S_i)$$

where S_i is the state to which O_i is applied.

Let us see that if the operator O_i has preconditions $P \subseteq Bp^*(S_i)$, then the corresponding operator $\gamma_1(O_i)$ has preconditions $P' \subseteq f_o(S'_i)$, and vice versa.

The condition

$$P \subseteq BP^*(S_i) \tag{5.16}$$

is equivalent to

$$P' \subseteq f_o(\ \mathbf{C}_R(S_i)\ )$$

as $P = P'$. As before, since both members are $\subseteq Lo$, this relation holds *iff*

$$P' \subseteq f_o(\ \mathbf{C}_R(S_i)\ ) \cup D$$

for any $D \subseteq \Delta$. Applying the property 5.12, we obtain

$$P' \subseteq f_o(\, \mathbf{C}_R(S_i) \cup D \,)$$

By choosing $D = (\, \Omega_1(S_i) \setminus \mathbf{C}_R(S_i) \,)$, the previous expression is equivalent to

$$P' \subseteq f_o(\Omega_1(S_i)) = f_o(S_i') \tag{5.17}$$

Since it results that 5.16 holds *iff* 5.17 holds, then we can conclude that if the operator O_i has preconditions $P \subseteq BP^*(S_i)$, then the corresponding operator $\gamma_1(O_i)$ has preconditions $P' \subseteq f_o(S_i')$, and vice versa.

Since both of the conditions 5.2 and 5.3 are equivalent for each pair of sequences Z and Z', then Z solves $\mathcal{P}$ *iff* Z' solves $\mathcal{P}'$.

Hence, the problem $\mathcal{P}$ is equivalent to $\mathcal{P}'$.

□

5.3.3 A2

The second transformation algorithm A2 has been designed to effect the integration of the group of rules Ro_a, corresponding to the second step of the process of integration of the outer extension function

$$BP^* = \mathbf{C}_{Ro_b} \circ \mathbf{C}_{Ro_a} \circ \mathbf{C}_{R_i}$$

The set Ro_a can be *any* subset of the set Ro, which is acyclic, as derived from the interpretation function $\mathcal{I}(\,)$ of the BP algorithm applying the method used in the proof of the theorem of Composed Closure.

As mentioned before, the properties of the boolean functions adopted by the interpretation $\mathcal{I}(\,)$ are reflected by the properties of the corresponding inference rules of Ro.

So, for example, let $w \in (Lo \setminus Li)$ have the interpretation

$$\mathcal{I}(w) = \neg\,(e_1 \vee e_2)$$

where $e_1, e_2 \in Li$. (Notice that this corresponds to the definition of interpretation for the *default* belief w =SUND(e), with $e_1 = e$ and e_2 =NOT(e), for the language

$<WFE>$ adopted by the belief system described in Chapter 4.)

This association will be transformed into the rule r of the set Ro

$$r) \quad \neg(e_1 \vee e_2) \vdash w$$

As a matter of fact, the set Ro_a is built by collecting *all* the rules of the set Ro which are in the form

$$r) \quad \neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_k \quad \vdash \quad c$$

with $q_i \in Li$ and $c \in (Lo \setminus Li)$.

A2 assumes these rules to be derived from the interpretation of a *default* belief 'c', such that the terms

$$\{q_1, \ldots, q_k\}$$

are subject to a relation of *restriction* which imposes their mutual exclusivity.

As we shall see in the following paragraphs, this assumption simplifies the transformation which A2 needs to carry out on the set of operators Op of the original problem $\mathcal{P}$ to obtain the new set of operators $Op' = \Gamma_2(Op)$, integrating $\mathbf{C}_{Ro_b}$.

A2

Let $\mathcal{P} = (I, Op, G, BP^*)$ be a BP-planning problem such that $BP^* = \mathbf{C}_{R_b} \circ \mathbf{C}_{R_a}$, where the sets of inference rules R_b and R_a have only consequences belonging to $Lo \setminus Li$, and R_a contains only rules in the form

$$\begin{array}{lll} r_1) & \neg(q_{11} \vee \ldots \vee q_{1k_1}) & \vdash c_1 \\ r_2) & \neg(q_{21} \vee \ldots \vee q_{2k_2}) & \vdash c_2 \\ \vdots & \vdots & \vdots \\ r_n) & \neg(q_{n1} \vee \ldots \vee q_{nk_n}) & \vdash c_n \end{array}$$

with $q_{ij} \in Li$ [18] and $c_x \neq c_y$ for any $x, y \in \{1, \ldots n\}$, $x \neq y$ (i.e., the rules have all different consequences).

If:

1. $\forall r_i \in R_a$, there exists a *restriction* on Lo such that the expressions of the

[18] Notice that this implies that the set R_a is *acyclic*.

set $\{q_{i1}, \ldots, q_{ik_i}\}$ result to be *mutually exclusive* [19],

then the algorithm A2, consisting of the application of the procedure Γ_2, transforms $\mathcal{P}$ into an equivalent BP-planning problem $\mathcal{P}' = (I', Op', G', BP_1^*)$, where:

- $I' = \mathbf{C}_{R_a}(I)$
- $Op' = \Gamma_2(Op)$
- $G' = G$
- $BP_1^* = \mathbf{C}_{R_b}$

The role of the transformation algorithm A2 consists of integrating the set of rules R_a into the BP-planning problem, producing a new problem with a simpler outer extension function $BP_1^* = \mathbf{C}_{R_b}$.

The idea which lies behind the method of integration adopted by A2 is the same idea upon which the definition of the algorithm A1 relies, namely, that of 'completing' the initial state I and the effects of the operators in Op with *all* the outer propositions introduced by the closure function $\mathbf{C}_{R_a}$.

However, it is already possible to recognize that A2 will constitute a 'simplified' version of A1, as the transformation Ω_1 has been substituted with the simple closure $\mathbf{C}_{R_a}$. This avoids the introduction of new expressions not belonging to the outer language Lo (the set Δ), and the definitions of the new inner and outer languages for the new BP-planning problem resulting from A2 will be

$$\begin{aligned} Li' &= \mathbf{C}_{R_a}(Li) \\ Lo' &= Lo \end{aligned}$$

The proof of the correctness of A2 is based on the specific properties of the transformation Γ_2, which will be defined in the next paragraphs; these properties are analogous to those of the function Γ_1 for A1 (see proposition 5.13).

However, as a consequence of the assumption 1., the transformations which A2 needs to carry out on each operator $O \in Op$ to produce the new set Op' are much simpler.

In fact, the operators of the given BP-planning problem $\mathcal{P}$ have been supposed to be *coherent* (see page 124), that is, to produce a coherent state whenever the state to which they are applied is coherent.

[19] This condition could be seen as requiring the rules to be *asynergistic*, in the sense given to this term by Chapman in [20].

Because of this, if the propositions $\{q_1, \ldots, q_k\}$ are mutually exclusive, only *one* of them can appear in the add list A, and only — a different — one can appear in the delete list D.

Moreover, the default belief c, belonging to $Lo \setminus Li$, is not allowed to be used in A or D, which must contain only inner expressions (by definition of BP-planning problem). Hence, the deletion of a certain q_i not mirrored by the addition of another $q_j \neq q_i$ will cause the default 'c' to be evaluated *True* in the state following the application of the operator.

Taking these considerations into account, it is possible to specify the algorithm for the transformation Γ_2.

Γ_2

The algorithm calculating the function Γ_2 consists of 'extending' each operator $O = (P, A, D) \in Op$ into a new operator $O' = (P', A', D') = \gamma_2(O)$ such that

- $P' = P$
- $A' = A \cup A^*$
- $D' = D \cup D^*$

where A^*, D^*, containing only expressions of the set $\{c_1, c_2, \ldots, c_n\} \subseteq (Lo \setminus Li)$, are obtained as described by the following steps:

1. let $A^* = \emptyset$, $D^* = \emptyset$;
2. $\forall a \in A$ and for each rule $r \in R_a$ containing a in its premisses

$$r) \quad \neg(q_1 \vee q_2 \vee \ldots \vee a \vee \ldots \vee q_k) \vdash c$$

   such that $\{q_1, q_2, \ldots, q_k\} \cap D = \emptyset$, add '$c$' to D^*, if it was not already present;
3. $\forall d \in D$ and for each $r \in R_a$ containing d in its premisses

$$r) \quad \neg(q_1 \vee q_2 \vee \ldots \vee d \vee \ldots \vee q_k) \vdash c$$

   such that $\{q_1, q_2, \ldots, q_k\} \cap A = \emptyset$, add '$c$' to A^*, if it was not already present.

Notice that γ_2 is 1–1, and it does not introduce conditional effects in the new set of operators. Moreover, it should be noticed that this algorithm can be easily extended to operators which contain *conditional effects*.

In fact, the effects of A2 on an operator $O = (P, A, D)$ simply consist of extending the add and delete lists A and D by adding to them two sets, A^* and D^* respectively, determined through the set of inference rules. The same extension can be carried out on *each* conditional effect $e_i = (P_i, A_i, D_i)$ of a list $[e_1, \ldots, e_n]$ present in the operator.

The properties valid for the transformation of an operator containing a *single* effect (P, A, D) will also be valid for the transformation of an operator containing a *list* of effects. This is due to the fact that the extension lists of the effects are totally *independent* from each other, and so are the changes they produce on the state at the moment of the application of the operator.

The formal proof for the correctness of this algorithm is given in the following section.

5.3.4 Correctness of A2

Given two set of rules R_b,R_a which satisfy the requirements specified for the application of A2, the following Lemma formally proves the validity, for the function $\gamma_2(\,)$, of the property analogous to 5.13, which concerned the transformation $\gamma_1(\,)$:

Lemma 5.1 *Given two coherent states S_t and $\mathbf{C}_{R_a}(S_t)$ and two operators $O = (P, A, D) \in Op$ and $O' = \gamma_2(O)$, with O coherent and such that $A \cap D = \emptyset$, $D \subseteq P$* [20]*, then the application of O, O', respectively, to the states S_t and $\mathbf{C}_{R_a}(S_t)$*

$$\begin{array}{ccc} S_t & \xrightarrow{O} & S_{t+1} \\ \downarrow & & \\ \mathbf{C}_{R_a}(S_t) & \xrightarrow{\gamma_2(O)} & S'_{t+1} \end{array} \tag{5.18}$$

produces two coherent states S_{t+1} and S'_{t+1} such that $S'_{t+1} = \mathbf{C}_{R_a}(S_{t+1})$.

Proof Since S_{t+1} and S'_{t+1} are obtained from the states S_t and $\mathbf{C}_{R_a}(S_t)$ applying O and O', then

$$\begin{aligned} S_{t+1} &= (A \cup S_t) \setminus D \\ S'_{t+1} &= (A' \cup \mathbf{C}_{R_a}(S_t)) \setminus D' \end{aligned}$$

[20] The condition $D \subseteq P$ means that, before deleting a proposition d, an operator in Op must verify that d is actually present in the state.

with $A' = A \cup A^*$, $D' = D \cup D^*$ [21]. Since $A \cap D = \emptyset$, the expression $(A \cup S_t) \setminus D$ is equivalent to $A \cup (S_t \setminus D)$. Moreover, because of the way A' and D' are built in A2, from $A \cap D = \emptyset$ it also follows $A' \cap D' = \emptyset$.

In order to show that $S'_{t+1} = \mathbf{C}_{R_a}(S_{t+1})$, we need to prove the two following relations:

a) $\mathbf{C}_{R_a}(S_{t+1}) \subseteq S'_{t+1}$

b) $S'_{t+1} \subseteq \mathbf{C}_{R_a}(S_{t+1})$

a) Let $c \in \mathbf{C}_{R_a}(S_{t+1})$; we need to prove that $c \in S'_{t+1}$. The analysis can be divided into two cases: 1) $c \in S_{t+1}$ or 2) $c \notin S_{t+1}$.

1) If $c \in S_{t+1} = A \cup S_t \setminus D$, then $c \in Li$, as S_t and the list A (and D) contain only elements of the inner language. Since D^* contains only elements of $Lo \setminus Li$, then $c \notin D^*$. Therefore,

$$\begin{aligned} c \in (A \cup S_t \setminus D) \setminus D^* &= A \cup S_t \setminus (D \cup D^*) \\ &\subseteq (A \cup A^*) \cup \mathbf{C}_{R_a}(S_t) \setminus (D \cup D^*) = S'_{t+1} \end{aligned}$$

(Notice that $S_t \subseteq \mathbf{C}_{R_a}(S_t)$ because of property 3.4). Hence, $c \in S'_{t+1}$.

2) Suppose now $c \notin S_{t+1}$, even though $c \in \mathbf{C}_{R_a}(S_{t+1})$. If this is the case, then the presence of c in $\mathbf{C}_{R_a}(S_{t+1})$ must be originated by a rule r in R_a

$$r) \quad \neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_k \vdash c$$

such that none of the premisses in $Pr = \{q_1, q_2, \ldots, q_k\}$ belongs to $S_{t+1} = A \cup S_t \setminus D$. Since the consequences of all the rules belong to $Lo \setminus Li$, then $c \notin D$, which contains only elements of Li. Again, there are two possible cases to be considered: either none of the premisses in Pr belongs to D, or (at most) one premiss $d \in Pr$ belongs to D.

In the first case, since $Pr \cap S_{t+1} = \emptyset$ and $Pr \cap (A \cup S_t) = \emptyset$, then $Pr \cap S_t = \emptyset$; hence, $c \in \mathbf{C}_{R_a}(S_t)$. For a given c, only one rule r of R_a can have c as consequence (all the rules in R must have different consequences); hence, c cannot have been

[21] Notice that O is applicable to S_{t+1} *iff* O' is applicable to $\mathbf{C}_{Ro_a}(S_t)$. This will be shown in the next theorem ('A2-Equivalence'). If both of O and O' are not applicable, then the Lemma clearly holds, as the two initial states are left unchanged.

added to D^* by A2, as this would require one element of the premisses in Pr to be in the add list A, whereas $Pr \cap (A \cup S_t) = \emptyset$. On the other hand, if $c \notin D^*$, then $c \notin D'$, hence $c \in S'_{t+1}$, as $c \in \mathbf{C}_{R_a}(S_t)$.

In the second case, $\exists d \in Pr$ such that $d \in D$. All the premisses in $Pr \setminus D$ cannot belong to A, as none of the premisses in Pr belongs to $A \cup S_t \setminus D$. Hence, $Pr \cap A = \emptyset$ (since $A \cap D = \emptyset$); this forces A2 to add c to the set A^*, which implies $c \in S'_{t+1}$ (as $A^* \cap D^* = \emptyset$).

b) Let $s \in S'_{t+1}$; we need to prove that $s \in \mathbf{C}_{R_a}(S_{t+1})$. Since $s \in S'_{t+1} = (A \cup A^*) \cup \mathbf{C}_{R_a}(S_t) \setminus (D \cup D^*)$, then $s \notin (D \cup D^*)$. The analysis can be divided into the following cases: 1) $s \in A$; 2) $s \in A^*$ (and $s \notin A$); 3) $s \in \mathbf{C}_{R_a}(S_t)$.

1) If $s \in A$, then $s \in A \cup S_t \setminus D$, which is a subset of its closure $\mathbf{C}_{R_a}(A \cup S_t \setminus D)$ (property 3.4). Hence, $c \in \mathbf{C}_{R_a}(A \cup S_t \setminus D) = \mathbf{C}_{R_a}(S_{t+1})$.

2) Consider $s \in A^*$ (and $s \notin A$). The presence of s in A^* must have been originated by A2 because of the existence of a rule r in R_a

$$r) \quad \neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_k \vdash s$$

such that (at most[22]) one of the premisses in $Pr = \{q_1, q_2, \ldots, q_k\}$ belongs to D, under the condition that $Pr \cap A = \emptyset$. If this is the case, then the considered operator O deletes one of the *mutually exclusive* expressions of Pr without adding another one (notice that $s \notin A$).

Since the state S_t is coherent (by hypothesis), *none* of the rest of mutually exclusive premisses $Pr \setminus D$ can be present in the state S_t. But this implies that none of these elements is in S_{t+1} either, as $Pr \cap A = \emptyset$. Hence, $Pr \cap S_{t+1} = \emptyset$; therefore, $s \in \mathbf{C}_{R_a}(S_{t+1})$, as $r \in R_a$.

3) Consider $s \in \mathbf{C}_{R_a}(S_t)$. If $s \in S_t$, then it follows immediately that $s \in A \cup S_t \setminus D$, from which $s \in \mathbf{C}_{R_a}(A \cup S_t \setminus D) = \mathbf{C}_{R_a}(S_{t+1})$.

On the other hand, if $s \notin S_t$ but $s \in \mathbf{C}_{R_a}(S_t)$, then there must be a rule r of R_a

$$r) \quad \neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_k \vdash s$$

[22] Because of the assumption of coherence of the operators.

such that none of the premisses in $Pr = \{p, q, \ldots, r\}$ belongs to S_t. Again, there are two possible cases: $Pr \cap A \neq \emptyset$, and $Pr \cap A = \emptyset$.

The first case, in which $\exists\, a \in Pr$ such that $a \in A$, results to be contradictory.

In fact, the assumptions of this Lemma specify that the operator O must be such that 1) D is a subset of the preconditions of O, and 2) O is applicable to the current state S_t. According to the definition of correct plan for a BP-planning problem, O is applicable to the state S_t iff its precondition list P is a subset of the outer extension $BP^*(S_t)$ (see condition 5.2).

But $BP^* = \mathbf{C}_{R_b} \circ \mathbf{C}_{R_a}$, and R_b, R_a are such that their consequences belong to $Lo \setminus Li$. Hence, since $D \subseteq Li$, if $\exists d \in D$, in order for O to be applicable to S_t it must be $d \in S_t$ (for hypothesis, $D \subseteq P$).

Since *none* of the premisses in Pr belongs to the state S_t, then $\nexists d \in D$ such that $d \in Pr$, i.e., $D \cap Pr = \emptyset$. Because of the way D^* is built by A2, if $\exists a \in A : a \in Pr$ and $D \cap Pr = \emptyset$ then s must have been added to D^*, which contradicts the fact that $s \notin (D \cup D^*)$, as mentioned at the beginning of case *b*).

Being the first case contradictory, it must be true that $Pr \cap A = \emptyset$. If none of the elements of Pr belongs to $A \cup S_t$, none of these elements appears in $A \cup S_t \setminus D$, either. Therefore,

$$s \in \mathbf{C}_{R_a}(A \cup S_t \setminus D) = \mathbf{C}_{R_a}(S_{t+1})$$

from which $s \in \mathbf{C}_{R_a}(S_{t+1})$.

Finally, after having proved that $S'_{t+1} = \mathbf{C}_{R_a}(S_{t+1})$, we need to show that the two states S_{t+1}, S'_{t+1} are coherent. S_{t+1} is coherent as obtained from S_t through O, which is coherent. But if $S'_{t+1} = \mathbf{C}_{R_a}(S_{t+1})$ and S_{t+1} is coherent, then also S'_{t+1} is coherent, as the rules in R_a cannot 'introduce' incoherence.

□

The following Theorem uses this Lemma to prove formally the equivalence between the two BP-planning problems $\mathcal{P}$ and $\mathcal{P}'$, where $\mathcal{P}'$ is the result of the transformation performed by the algorithm A2 on $\mathcal{P}$:

Theorem 5.3 (A2-equivalence) *Let $\mathcal{P} = (I, Op, G, BP^*)$ be a BP-planning problem, with $BP^* = f_o \circ f_i$, where:*

- *$f_i = \mathbf{C}_{R_a}$, $f_o = \mathbf{C}_{R_b}$, and the sets of inference rules R_a, R_b satisfy the requirements for the application of the algorithm A2;*

- $\forall O \in Op$, $O = (P, A, D)$ *is coherent and* $A \cap D = \emptyset$, $D \subseteq P$.

The new BP-planning problem $\mathcal{P}' = (\ \mathbf{C}_{R_a}(I),\ \Gamma_2(Op),\ G,\ f_o\ )$ *produced by A2 is* equivalent *to* $\mathcal{P}$.

Proof The proof of this theorem is a simplified version of the proof for A1. Consider a sequence of steps $Z{=}<O_1, O_2, \ldots, O_n>$ with $O_i \in Op$, and associate it with the sequence $Z'{=}<\gamma_2(O_1), \ldots, \gamma_2(O_n)>$ of operators $\in \Gamma_2(Op)$. This association represents a *bijection*, as the transformation γ_2 is *1–1*, and each operator in $\Gamma_2(Op)$ has been generated by one (and only one) operator in Op.

We need to prove that if Z solves $\mathcal{P}$, then Z' solves $\mathcal{P}'$, and vice versa.

Let

$$I \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \ldots \xrightarrow{O_n} S_n = F$$

be the sequence of states obtained applying the plan Z, and

$$\mathbf{C}_{R_a}(I) \xrightarrow{\gamma_2(O_1)} S'_1 \xrightarrow{\gamma_2(O_2)} \ldots \xrightarrow{\gamma_2(O_n)} S'_n = F'$$

the sequence generated by Z'.

Because of Lemma 5.1, since $\mathbf{C}_{R_a}(I) \xrightarrow{\gamma_2(O_1)} S'_1$, then $S'_1 = \mathbf{C}_{R_a}(S_1)$ (and S'_1 is coherent). Analogously, $S'_2 = \mathbf{C}_{R_a}(S_2)$, $S'_3 = \mathbf{C}_{R_a}(S_3)$, etc.

Let us see that if $S_n = F$ is a *final* state for $\mathcal{P}$ (i.e. it contains the goal set G) then $S'_n = F' = \mathbf{C}_{R_a}(S_n)$ is a final state for $\mathcal{P}'$, and vice versa.

F is final state for $\mathcal{P}$ *iff*

$$\forall g \in G,\ \ g \in BP^*(F) \tag{5.19}$$

by definition of correct plan solution of a BP-planning problem (see equation 5.3). This is equivalent to

$$G \subseteq f_o(\ \mathbf{C}_{R_a}(F)\ )$$

But since $G' = G$, this condition is the same as

$$G' \subseteq f_o(\ \mathbf{C}_{R_a}(F)\ )$$

which becomes directly

$$G' \subseteq f_o(F') \tag{5.20}$$

This condition expresses the fact that F' is a final state for the new problem $\mathcal{P}'$. Since it results that 5.19 holds *iff* 5.20 holds, then we have proved that if $S_n = F$

is a *final* state for $\mathcal{P}$ then $S'_n = F'$ is a final state for $\mathcal{P}'$, and vice versa.

The second part of the proof concerns the condition 5.2, requiring that for every operator O_i used in a solution, if P is its precondition list, then

$$P \subseteq BP^*(S_i)$$

where S_i is the state to which O_i is applied.

Let us see that if the operator O_i has preconditions $P \subseteq Bp^*(S_i)$, (i.e. O_i is applicable to the state S_i) then the corresponding operator $\gamma_2(O_i)$ has preconditions $P' \subseteq f_o(S'_i)$ (i.e. $\gamma_2(A_i)$ is applicable to S'_i), and vice versa.

The condition

$$P \subseteq BP^*(S_i) \tag{5.21}$$

is equivalent to

$$P' \subseteq f_o(\ \mathbf{C}_R(S_i)\ )$$

as $P = P'$; but this is equivalent to

$$P' \subseteq f_o(S'_i) \tag{5.22}$$

Since it results that 5.21 holds *iff* 5.22 holds, then we can conclude that if the operator O_i has preconditions $P \subseteq BP^*(S_i)$, then the corresponding operator $\gamma_2(O_i)$ has preconditions $P' \subseteq f_o(S'_i)$, and vice versa.

Since both of the conditions 5.2 and 5.3 are equivalent for each pair of sequences Z and Z', then Z solves $\mathcal{P}$ *iff* Z' solves $\mathcal{P}'$. Hence, the problem $\mathcal{P}$ is equivalent to $\mathcal{P}'$.

□

5.3.5 The integration process

The entire process of integration of a BP algorithm into a planning system is based on the application of algorithms A1 and A2 in conjunction with the Theorem 5.1 (of the Composed Closure).

In fact, the procedure described in the proof of the latter allows any outer extension function $BP^*(\,)$ (containing, in an extended form, the same 'information'

of the $BP()$ function) to be written as the composition of two closures

$$BP^* = \mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}$$

Since the set Ro results to be *acyclic*, it can be further divided into two sets, Ro_a and Ro_b, which are integrated separately through the application of A1 and A2, respectively.

Thus, the whole integration process can be represented by the sequence of problem transformations shown by the diagram in Figure 5.1.

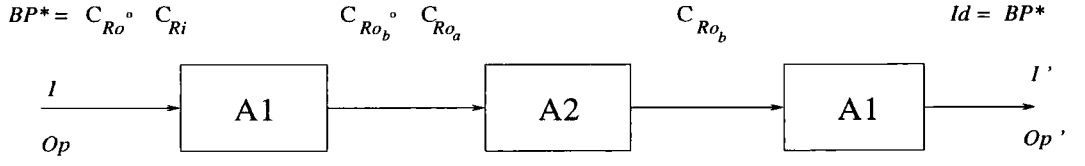


Figure 5.1: Integration process.

Each phase of the process produces a new BP-planning problem equivalent to the previous one, until the last problem is solvable by a standard planning system. Nevertheless, there are two points which should be noticed.

First of all, although the input and output of each 'block' consist both of a BP-planning problem, the set of operators produced by A1 contains *conditional effects*, whereas the set in input to it should contain exclusively non-conditional (STRIPS-like) operators (P, A, D). To allow conditional effects to be pre-processed, A1 will have to be applied to each triple (P_i, A_i, Di) of each conditional operator in input Moreover, A1 will have to be applied more than once in the first phase if the initial set Ri is not acyclic and needs to be split into separate blocks.

Secondly, the applicability of the process of integration relies upon a set of hypotheses and restrictions made during the introduction and formalization of the indirect method of integration, which, for clarity, have been summarized below.

1. The first obvious assumption consists of the fact that the problem to be solved must satisfy the requirements necessary to be classified as a BP-planning problem. That is, there must be two languages, $Li \subseteq Lo$ (inner and outer), an outer extension function $BP^*() = \mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}$, derived from a BP algorithm which calculates an interpretation $\mathcal{I}()$ of Lo on Li using a set of rules Ri on Li, an initial state $I \subseteq Li$, a set of goals $G \subseteq Lo$ and a set of operators Op with specific characteristics (see definition at page 122).

2. The inference rules of Ri are in the form $p \vdash c$ (with $p, c \in Lo$) and Ri

is *acyclic*. In alternative, if Ri is not already acyclic, it must be possible to divide it into an ordered sequence of acyclic blocks, such that the consequences of each block do not appear as premisses of the previous blocks (this hypothesis is required for the applicability of A1).

3. The rules in Ro, the set of rules obtained from $\mathcal{I}()$ through the method described in Theorem 5.1, are in one of the two following forms:

   $a)$ $\neg(q_1 \vee q_2 \vee \ldots \vee q_k) \vdash c$

   $b)$ $p \vdash c$

   with $q_i \in Li$, $p \in Lo, c \in (Lo \setminus Li)$ and such that: i) 'grounding' p makes c grounded too, and ii) the rules of type a) are obtained from the interpretation of a *default* belief $\mathcal{I}(c) = \neg(q_1 \vee \ldots \vee q_k)$, which imposes a restriction of mutual exclusivity and exhaustivity on the terms $\{q_1, \ldots, q_k, c\}$ [23].

4. Each operators $O \in Op$ must be *coherent*, and such that if $O = (P, A, D)$, then $A \cap D = \emptyset$ and $D \subseteq P$ (these hypotheses are necessary for the applicability of A2, although the set of rules in Ro of type b) is integrated by A1).

As already discussed, these assumptions guarantee that the application of A1 and A2 will produce the expected results, but their *necessity* is only contingent to the specific adoption of such algorithms, and can be avoided with the development of more sophisticated procedures of integration.

Moreover, as it will be shown in the next chapter, most of these hypotheses are quite 'naturally' satisfied when considering a real domain example.

Any belief system defined according to the paradigm described in Chapter 3 and satisfying these hypotheses can undergo the process of indirect integration, and become part of an automatic system for the solution of BP-planning problems.

However, involved in the global process of integration, there is a more 'practical' issue which has not been analysed properly, yet. It concerns the *finiteness* of the process. The presence of infinite sets in the problem can result from the adoption of infinite languages Lo and Li. The non-finiteness of the languages has an impact on the whole process of integration.

[23] Notice that the sets Ro_b and Ro_a will contain the collection of all the rules of Ro in the form, respectively, b) and a).

In fact, for example, the interpretation function $\mathcal{I}(\,)$ must associate *each* expression of the domain $Lo \setminus Li$ to a boolean function of inner propositions. If the set $Lo \setminus Li$ is finite, this association is representable simply with a finite list of pairs, and can be transformed into the equivalent (finite) set of rules Ro, using the method described in Theorem 5.1. But if the set $Lo \setminus Li$ contains an infinite number of expressions, the list of associations results to be infinite: hence, the process of conversion of this list into the set of rules Ro will never terminate.

In order to guarantee the finiteness of the process of integration, it is necessary at least to give a finite *representation* of the infinite sets; this can be done only with the use of expressions containing *typed variables*, which can be substituted with values taken from specific sets containing infinite elements.

As a matter of facts, this is what has been done, implicitly, to define the inference rules of Ri for the belief system described in Chapter 4. For example, the inference rule

$$i_1) \qquad \mathrm{SR}(e) \vdash e$$

(with $e :: <Ei>$) represents finitely an infinite set of rules, since the production $<Ei>$ can generate an infinite list of events $\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_n, \ldots$.

The introduction of variables in the definition of the interpretation $\mathcal{I}(\,)$ might lead to the presence of variables in the result of the closure $\mathbf{C}_{Ro}$. In fact, consider, for example, the definition of $\mathcal{I}(\,)$ given in Chapter 4, which contains (vice versa) the following association:

$$\mathcal{I}(\mathrm{HUNK}(\mathrm{SUP}(e, e'))) = B(\mathrm{HUNK}(e)) \vee B(\mathrm{HUNK}(e'))$$

with $e, e' :: <Ei>$. The conversion of this association into inference rules will produce the two rules of Ro

$$\begin{aligned} r_i) \quad & \mathrm{HUNK}(e) \vdash \mathrm{HUNK}(\mathrm{SUP}(e, x)) \\ r_{i+1}) \quad & \mathrm{HUNK}(e) \vdash \mathrm{HUNK}(\mathrm{SUP}(x, e)) \end{aligned}$$

where $e, x :: <Ei>$.

Whatever the expression in the premisses is matched with, the variable x of the consequences will always be *uninstantiated.* Hence, if any of these rules is applied during the calculation of the closure $\mathbf{C}_{Ro_b}(I)$, performed by A1 in the last phase of the integration process, the resulting initial state of the BP-planning problem will contain expressions with variables.

The presence of variables in the propositions of a state may not always be

allowed by a planning system, especially when these variables represent an infinite set of expressions. To eliminate this problem without changing the definition of the interpretation which caused the introduction of variables, it might be necessary to restrict to a finite set the domain of each variable, and substitute each expression of the state containing variables with the set of expressions obtained by instantiating the variables in every possible way.

However, an *arbitrary* restriction of the variable domain could corrupt the correct functioning of a system. A more suitable approach consists of allowing a 'lazy evaluation' of the infinite sets of expressions. This means considering the domain of the variables to be 'virtually' infinite: since only a small portion of these values will actually be used during the planning process, it will be sufficient to instantiate the variables, at 'run time', with only those values which will be strictly needed for the specific computation required. It should be noticed that the idea of restricting the domain of the variables to the sets of values needed for the specific problem considered can be used to optimise the performance of planning systems in general, even when the language adopted is finite. In fact, even if the number of possible instances of a variable is finite, there can still be a large number of possibilities, and the restriction of the domains to smaller sets can notably reduce the search space examined during the planning process.

As next chapter will illustrate (Section 6.3), the same idea has been adopted for the actual implementation of the integration of a specific belief system, realised according to the indirect (or 'preprocessing') method of integration described so far.

5.3.6 Conclusions

The aim of this chapter was to analyse the feasibility and method of realization of the integration between a belief system implemented according to the paradigm described in Chapter 3 and a generic planning system.

The theoretical analysis has shown that, under the assumptions summarized in the previous section, any BP algorithm can be integrated into any standard planning system. Since the system described in Chapter 4 has been defined according to the specific requirements for a BP algorithm and satisfies the necessary hypotheses concerning the rules in Ri and Ro, the results of this chapter confirm its suitability for integration (preprocessing) into a wide range of planners.

Although the indirect method of integration described can require the 'hand-coding' of the inference rules in Ro, derived from $\mathcal{I}(\,)$, it presents the big advantage

of freeing the programmer from any modification of the code of the 'host' planning system, shifting the human intervention from the correction of a pre-existent piece of software to the development of a separate 'integrator' system, which can be designed, implemented and modified as an independent module.

The next chapter will illustrate, as an example, the implementation and integration of a belief system for persuasive discourse planning, which has been developed and preprocessed according to the two described paradigms.

Chapter 6

A Persuasive Discourse Planning System

The aims of this final chapter are: 1) to show how, in practice, the integration of a belief system for discourse planning defined according to the paradigm of Chapter 3 has been realized, and 2) to illustrate the results of the system implemented, which constitutes an example of *persuasive discourse planner.*

The first section describes the specific BP algorithm adopted by the belief system. The second section gives a brief summary of the high-level steps of the algorithm for the integration, whereas the third one contains a more detailed description of the practical aspects concerning the actual implementation of the system, and introduces the set of operators which have been given to the planner to solve the BP-planning problems.

The fourth and last section illustrates some examples of persuasive discourse plans, produced in different belief situations and to achieve different goals.

6.1 The Integrated belief system

The belief system adopted for the integration has been realized according to the same theoretical framework upon which the model described in Chapter 3 relies. As such, it can be described as a triple (Lo, Li, A), where A is the BP algorithm which calculates the interpretation of the (outer) language Lo on the (inner) language of beliefs Li.

The following paragraphs introduce and discuss the formal features of these languages and the other characteristics of the BP algorithm.

6.1.1 The outer language

The outer language *Lo* recognized by the system consists of a simplified version of the two-level nesting $<WFE>$ outer language previously formalized. Its syntax is defined by the following BNF production rules, where $<WFE1>$ should be used as initial symbol:

$$
\begin{aligned}
<WFE1> &::= <Pred>(<A>) \mid \text{NOT}(<Pred>(<A>)) \\
<Pred> &::= \text{SBEL} \mid \text{SUND} \mid \text{HBEL} \mid \text{HUND} \mid \text{HUNK} \\
<A> &::= <E> \mid \text{NOT}(<E>) \\
<E> &::= \text{SR}(<Es>) \mid \text{HR}(<Es>) \mid <Es> \\
<Es> &::= \text{SUP}(<En>, <En>) \mid <Ei> \\
<En> &::= <Es> \mid \text{NOT}(<Es>) \\
<Ei> &::= \text{E}_0 \mid \text{E}_1 \mid \text{E}_2 \mid \text{E}_3 \mid \ldots \mid \text{E}_n \mid \ldots
\end{aligned}
$$

There are three main differences which distinguish this syntax from the original $<WFE>$:

1. $<WFE1>$ allows only one level of belief nesting instead of two;
2. the arguments of a SUP( ) expression must be either a SUP( ) expression, a primitive event $<Ei>$, or one of their negations, whereas in $<WFE>$ they could have been any expression generated by $<A>$;
3. the new definition of the production rule for the symbol $<E>$ eliminates the groups of SIEXP and HIEXP expressions (Speaker/Hearer Inductive EXPeriences).

As a consequence of the first limitation, all the propositions belonging to the hearer's model of the speaker's belief (such as HBEL(SBEL(E_1))) are not permitted. Nevertheless, it should be noted that beliefs such as HBEL(SR(E_2)) are still allowed, as they express the hearer's model of the speaker's real experience, and not of the speaker's beliefs.

The second simplification limits the range of support relationships actually expressible. In this syntax, for example, real-experience events in the form SR( ), HR( ) can not have any supporting event. This excludes expressions such as SUP(E_2,SR(E_3)) or SUP(HR(E_4),E_5).

Finally, because of the third simplification, the two HIEXP(s) and SIEXP(s) expressions, used in $<WFE>$ to indicate that a SUP( ) relationship s was the

result of *induction* based on a collection of repeated *causal* experiences, will now be represented by HR(s) and SR(s), respectively. Hence, in this context, a support event SUP( ) which was derived through the process of induction on multiple experiences will be treated like a *real* experience itself, representing an *unquestionable* belief resulting from the experiences of the subject.

Although the above language restricts the power of expressiveness of the original $<WFE>$, its syntax maintains intact the basic structure and characteristics, allowing the presence of non-limited levels of nesting for SUP( ) expressions, three-valued uncertainty (for both the speaker's and hearer's models), belief grounding through rationality or experience and identification of hearer *non-knowledge* via the adoption of a 'HUNK' predicate.

Therefore, the simplified language defined by $<WFE1>$, although developed mainly as part of an explanatory example, can be used to illustrate how the practical issues involved in the process of indirect integration of a belief system discourse-planning 'oriented' can be solved; moreover, as the final results will show, it still enables the description of realistic belief situations which generate the construction of sophisticated discourse plans.

6.1.2 The Interpretation

As described in Chapter 3, the function of interpretation $\mathcal{I}( )$ associates every outer language ($<WFE1>$ generated) expression with a boolean function defined over a subset of the outer language itself. The set of expressions obtained as the union of all the domains of the boolean functions will constitute the *inner* language.

The specific BP algorithm implemented effects this association through the application of the steps described below.

Let w be an expression of $Lo \leftarrow <WFE1>$; the boolean function associated to w is

$$\mathcal{I}(w) = check(w')$$

where w' and the function $check( )$ are defined as follows:

1. w' is obtained from w by applying, whenever possible, the following syntac-

tical transformations:

$$\begin{array}{lcl} \text{SUND(NOT}(e)) & \rightarrow & \text{SUND}(e) \\ \text{HUND(NOT}(e)) & \rightarrow & \text{HUND}(e) \\ \text{HUNK(NOT}(e)) & \rightarrow & \text{HUNK}(e) \\ \text{SBEL}(a) & \rightarrow & a \end{array}$$

for any string a and e such that the left-hand sides of the reductions are expressions of $<WFE1>$. It is important to notice that these simplifications should be applied to every part of the expression (i.e. even inside other predicates). For example, the expression NOT(SBEL(a)) should be reduced to NOT(a).

2. The function *check*( ) is defined as follows:

$$check(w') =$$

$$\begin{array}{ll} \neg B(\text{SR}(e)) & \text{if } w' \cong \text{NOT(SR}(e)); \\ B(w') & \text{if } w' :: <A>; \\ \neg check(x) & \text{if } w' \cong \text{NOT}(x); \\ B(w') & \text{if } w' \cong \text{HBEL(\_) or } w' \cong \text{HUND(\_)}; \\ \neg\,(\,B(e) \vee B(\text{NOT}(e))\,) & \text{if } w' \cong \text{SUND}(e); \\ \neg\,(\,B(\text{HBEL}(e)) \vee B(\text{HUND}(e)) \vee & \\ \quad B(\text{HBEL(NOT}(e)))\,) & \text{if } w' \cong \text{HUNK}(e); \end{array}$$

where the symbols '$\cong$', '::', '$\neg$' and '_' have the same meaning adopted in Chapter 4.

The only boolean functions which are worth noticing are those associated with the expressions NOT(SR(e)), SUND(e) and HUNK(e).

The first one assumes NOT(SR(e)) to be true iff the proposition is explicitly stated, and is identical to the corresponding interpretation defined for $<WFE>$.

The other two functions realize the mechanism of default belief for the two expressions SUND( ) and HUNK( ): if the current state does not explicitly contain the speaker's belief (or unbelief) towards an event (represented respectively by 'e' or 'NOT(e)'), then the system will assume the belief SUND(e) to hold. Similarly, the absence of any particular model of the hearer's attitude in relation to an event is equivalent to assuming that the hearer does *not know* the event (i.e. HUNK(e)).

It is interesting to note that the definition of the boolean function for NOT(SR(e))

forces any (outer) belief having the form SUND(SR(e)) to be evaluated *False* by the BP algorithm: in other words, this expression is considered contradictory, exactly like in the model for $<WFE>$.

It is not difficult to see that every possible $<WFE1>$ expression is taken into consideration by this algorithm.

6.1.3 The Inner language

The union of the domains of definition of all the boolean functions associated to the outer expressions constitutes the inner language Li (clearly a subset of the outer one) identified by this specific BP algorithm. As it can be seen from the definition of the function *check*( ), Li does not contain expressions such as SBEL(a), SUND(a), HUND(NOT(e)), HUNK(a), or NOT($<Pred>(a)$). In fact, it is easy to discover that the inner language identified by *check*( ) contains only the expressions generated by the three constructs $<A>$, HBEL($<A>$) and HUND($<E>$).

Moreover, since any expression in the form SUND(SR($<Es>$)) has been considered contradictory, the same assumption should be adopted for the symmetrical case HUND(HR($<Es>$)). In order to force these expressions to be always evaluated *False* it is sufficient to exclude them from in the inner language, so that they will never appear in the set of beliefs.

Summarising, the inner language can be generated simply by the two following production rules, where the initial symbol is $<WFF1>$, and $<A>$, $<Es>$ are defined as in $<WFE1>$:

$$\begin{aligned} <WFF1> &::= <A> \mid \text{HBEL}(<A>) \mid \text{HUND}(<Eh>) \\ <Eh> &::= \text{SR}(<Es>) \mid <Es> \end{aligned}$$

As a matter of fact, the expressions w in the form NOT(SR(e)), derivable from the symbol $<A>$, should not be considered as elements of the inner language $Li \leftarrow <WFF1>$, as they are associated, via $\mathcal{I}$( ), to a specific boolean function (namely, $\neg(B(\text{SR}(e)))$) which differ from the 'identity' function $B(w)$, to which *all* the $w \in Li$ are supposed to be associated.

Hence, the syntax $<WFF1>$ should be modified so as to contain all the expressions generated by $<A>$, but avoid the production of those in the form NOT(SR( )). However, for reasons of clarity, the previous syntax has been left unchanged, although the set of expressions having the mentioned form will be considered as belonging to $Lo \setminus Li$.

The Inference Rules

Having formally defined the inner language, it is now possible to introduce the set Ri of inference rules. These have been simply identified by selecting, amongst the rules defined for $<WFF>$ in Chapter 4, those which contained expressions admitted by the new restricted language $<WFF1>$.

As a result, only the five rules $\{i_1, i_3, i_5, i_7, i_{13}\}$ have been able to be included:

i_1) SR(e) $\vdash e$
i_2) HR(e) $\vdash e$, HBEL(HR(e))
i_3) NOT(HR(e)) $\vdash$ HBEL(NOT(HR(e)))
i_4) HBEL(SR(e)) $\vdash$ HBEL(e)
i_5) HBEL(HR(e)) $\vdash$ HBEL(e)

with $e :: <Es>$.

Notice that this list of inference rules can be considered complete, with respect to the set of possible rules with only one left hand side and containing exclusively inner language expressions.

For example, the five inference rules showed below, having single left hand side and containing only inner language expressions, have been considered but rejected:

- NOT(SR(e)) $\vdash$ NOT(e)
- NOT(HR(e)) $\vdash$ NOT(e)
- HBEL(NOT(SR(e))) $\vdash$ HBEL(NOT(e))
- HBEL(NOT(HR(e))) $\vdash$ HBEL(NOT(e))
- SUP(e_1, e_2) $\vdash$ SUP(NOT(e_2), NOT(e_1))

The incorrectness of the first four is obvious: believing that 'e' is not a real experience does not support the belief in NOT(e). The non applicability of the fifth one has been already discussed in Section 4.2.1.

Restrictions of $<WFF1>$

The interpretation $\mathcal{I}()$ contains two *default* belief mechanisms, defined for the expressions HUNK() and SUND(). Since $\mathcal{I}()$ will be converted — by the process of integration — into a set of inference rules Ro, the terms involved in the boolean functions associated to the default belief must be subject to a restriction of *mutual*

exclusivity [1].

The restrictions imposed on the inner language Li of the belief system consists of the two following boolean functions:

$$\begin{aligned}
\mathcal{U}_1) \quad & \neg \ (e \wedge \mathrm{NOT}(e)) \\
\mathcal{U}_2) \quad & \neg \ ((\mathrm{HBEL}(e) \wedge \mathrm{HBEL}(\mathrm{NOT}(e))) \vee \\
& \quad (\mathrm{HBEL}(e) \wedge \mathrm{HUND}(e)) \vee \\
& \quad (\mathrm{HUND}(e) \wedge \mathrm{HBEL}(\mathrm{NOT}(e))))
\end{aligned}$$

$\mathcal{U}_1 \mid_S$ and $\mathcal{U}_2 \mid_S$ must be always *True*, for any current (extended) state $S \subseteq Li$ of the system, for any string e admitted by the $<WFF1>$ syntax in those positions.

Similarly to the system described in Chapter 4, the effects of the restrictions imposed on Li group the expressions of Lo into two classes of sets such that the propositions belonging to the same set results to be mutually exclusive and exhaustive. The two classes of sets are as follows:

$$\begin{aligned}
S_1(e) &= \{e, \mathrm{NOT}(e), \mathrm{SUND}(e)\} \\
S_2(e) &= \{\mathrm{HBEL}(e), \mathrm{HBEL}(\mathrm{NOT}(e)), \mathrm{HUND}(e), \mathrm{HUNK}(e)\}
\end{aligned}$$

Notice that the operators which will be used by the planning system for the construction of discourse plans will be required to be *coherent* with these restrictions, i.e. to be such that their application to a coherent state will always produce a new coherent state.

6.2 Integration

The process of integration of the belief system described in the previous section is based on the indirect approach, consisting of the transformation of any given BP-planning problem $\mathcal{P}$ into an equivalent standard problem $\mathcal{P}'$ which can be solved normally.

The whole transformation requires the conversion of the interpretation function $\mathcal{I}()$ into a set Ro of inference rules; since $\mathcal{I}()$ has not been defined as an explicit list of associations, this preliminary conversion had to be performed 'by hand'.

[1] There is actually also a third default mechanism in $\mathcal{I}()$, defined for the expression NOT(SR(e)). However, its boolean function contains only one term; thus, there is no need for any further restrictions of mutual exclusivity, as this should be imposed on a ***single*** proposition.

The results of this initial phase consist of the following list of rules, constituting the set Ro:

$o_1)$ $\neg$ SR(e') $\vdash$ NOT(SR(e'))

$o_2)$ $\neg$ (e $\vee$ NOT(e)) $\vdash$ SUND(e), SUND(NOT(e))

$o_3)$ $\neg$ (HBEL(e) $\vee$ HUND(e) $\vee$ HBEL(NOT(e))) $\vdash$ HUNK(e), HUNK(NOT(e))

$o_4)$ a $\vdash$ SBEL(a), NOT(SUND(a)), NOT(SBEL(NOT(a))), NOT(SUND(NOT(a)))

$o_5)$ HBEL(a) $\vdash$ NOT(HBEL(NOT(a))), NOT(HUND(NOT(a))), NOT(HUNK(NOT(a))), NOT(HUND(a)), NOT(HUNK(a))

$o_6)$ HUND(e'') $\vdash$ HUND(NOT(e'')), NOT(HBEL(e'')), NOT(HUNK(e'')), NOT(HBEL(NOT(e''))), NOT(HUNK(NOT(e'')))

$o_7)$ SUND(e) $\vdash$ NOT(SBEL(e)), NOT(SBEL(NOT(e)))

$o_8)$ HUNK(e) $\vdash$ NOT(HBEL(e)), NOT(HBEL(NOT(e))), NOT(HUND(e)), NOT(HUND(NOT(e)))

with $e :: <E>$, $e' :: <Es>$,$e'' :: <Eh>$ and $a :: <A>$.

First of all, it should be noticed that the list has been divided into *three* acyclic blocks — $Ro_a = \{o_1\}$, $Ro_b = \{o_2, o_3\}$ and Ro_c containing all the remaining rules — which will be integrated separately and in the given order, using the algorithms A1 and A2.

Moreover, because of the way A1 and A2 work, after each integration, the inner language Li' of the *new* equivalent problem will contain also the expressions which appeared as consequences of the integrated block. In fact, as specified in the description of A1 and A2, $Li' = \Omega_1(Li)$ (in A1) and $Li' = \mathbf{C}_R(Li)$ (for A2).

This means that some of the expressions which were not elements of the inner language Li of $\mathcal{P}$ could be members of the new inner language Li' of $\mathcal{P}'$. Hence, these expressions, initially not allowed in the *premisses* of the rules of $\mathcal{P}$ which are being integrated[2] can appear in the premisses of the rules of the new problem, i.e., of the *following blocks.*

[2]The requirements for A1 and A2 specify that every rule of Ro must have premisses containing only inner language terms.

This happens, for example,in the case of the *default* expressions SUND(_) and HUNK(_), which are consequences of block Ro_b and premisses of Ro_c.

Let us explain how these inference rules can 'replace' the effects of the interpretation function $\mathcal{I}(\,)$.

Rule o_1 realizes the task previously performed by the function *check*( ) to calculate the truth value of the expression NOT(SR(e)), associated with the boolean function $\neg B(\text{SR}(e))$. The rule should be interpreted as follows: for each expression e' generable from $<Es>$, if the proposition SR(e') is *not* present in the state, then the expression NOT(SR(e)) should be *added* to the state.

Analogously, o_2 and o_3 replace the boolean functions used to calculate the truth of the *default* expressions SUND(e) and HUNK(e), respectively. For example, o_2 will add SUND(e) and SUND(NOT(e)) to the state for each expression $e :: <E>$ such that neither e nor NOT(e) are present in the state.

After completing the state with all the default expressions deduced from the *absence* of associated inner beliefs, the three rules o_4–o_6 will append all the 'outer' language expressions derivable from the inner beliefs currently present in the state.

Notice that the additions of the beliefs SBEL(a) in o_4 and HUND(NOT(e'')) in o_6 'substitute', respectively, the syntactical transformations SBEL(a) $\rightarrow a$ and HUND(NOT(e)) $\rightarrow$HUND(e).

Similarly, the two pairs of negated defaults added by o_5 and o_6 replace the analogous transformations for SUND(NOT(e)) and HUNK(NOT(e)).

Finally, o_7 and o_8 complete the state with all the outer beliefs which follow from the defaults previously added by o_2, o_3. It should be also pointed out that expressions like SUND(SR( )) or HUND(HR( )) — excluded from the inner language $<WFF1>$ — cannot be generated by any of these inference rules[3].

Having converted the interpretation $\mathcal{I}(\,)$ into the set of rules Ro, it is possible to rewrite the outer extension function of the BP as

$$BP^* = \mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}$$

where Ri is the set $\{i_1, \ldots, i_5\}$ of rules on Li introduced earlier in this section.

Notice that Ri needs to be divided into two separate blocks, not being already acyclic: in fact, the consequence HBEL(HR(e')) of i_2 constitutes the premiss of i_5.

Assuming to identify the two blocks composing Ri as Ri_1 and Ri_2 (which rules

[3]Notice that o_2 will never generate SUND(SR( )) because of the presence of o_1.

they will contain is not relevant, so long as $i_2 \in Ri_1$ and $i_5 \in Ri_2$), the entire integration process will consist of reducing the outer extension function

$$BP^* = Id \circ (\mathbf{C}_{Ro_c} \circ \mathbf{C}_{Ro_b} \circ \mathbf{C}_{Ro_a}) \circ (\mathbf{C}_{Ri_2} \circ \mathbf{C}_{Ri_1})$$

to the identity function Id through the following sequence of transformations:

$$\mathcal{P} \rightarrow A1(Ri_1) \rightarrow A1(Ri_2) \rightarrow A2(Ro_a) \rightarrow A2(Ro_b) \rightarrow A1(Ro_c) \rightarrow \mathcal{P}' \qquad (6.1)$$

The argument of each of the various applications of A1 and A2 represents the specific set of rules which is being integrated in that step.

The whole integration will consist of applying this sequence of transformations to any set of operators Op and initial state I[4], producing a resulting problem $\mathcal{P}' = (Op', I')$ which can be solved by a standard planning system.

Therefore, the resulting system will constitute an 'integrator' of BP-planning problems (specific for the BP algorithm adopted) producing in output the standard planning problems equivalent to the input ones.

The next section will explain the details of the implementation of this system, which has been realized by taking into account the characteristics of the specific 'host' planning system adopted.

Finally, the last result of this section consists of a theorem which guarantees the correctness of the conversion of the specific $\mathcal{I}(\,)$ adopted into the set of rules Ro: this is proved by showing that the function $\mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}(S)$ calculates *exactly* the outer extension $BP^*(S)$ for any given state S.

Theorem 6.1 *If $S \subseteq Li$ be a state, $Ri = \{i_1, \ldots, i_5\}$ and $Ro = \{o_1, \ldots, o_8\}$, then $(\mathbf{C}_{Ro} \circ \mathbf{C}_{Ri}(S)) = \{\omega \in Lo \mid BP(\omega, S) = True\}$.*

Proof In order to prove that $S^* = O$, let us see that $S^* \subseteq O$ and vice versa.

Part 1 ($S^* \subseteq O$): Let S be a subset of Li, and consider an element $s \in S^*$. We have to show that $s \in O$. There are two possible cases: either $s \in Li$ or $s \notin Li$.

If $s \in Li$, then either s was already in S, or it was added by a rule of Ri. In the first case, from $s \in S$ it follows $BP(s, S) = True$ (property 3.6. If s was added by Ri, then it will be evaluated *True* by the BP, because of the presence of the same

[4]Notice that the goal set G is always left unchanged by A1 and A2.

inference rules i_1–i_5 in the algorithm. Hence, if $s \in Li$, then $BP(s, S)$ is always true, and, therefore, $s \in O$.

Let us now consider $s \in S^*$ such that $s \notin Li$, i.e. $s \in (Lo \setminus Li)$. Since $s \in S^*$ but also $s \notin S$ (as it is not an inner expression) then it must have been added by one of the rules in Ro.

If it was added by o_1, then s=NOT(SR(e')), and SR(e') $\notin S$: thus, the boolean function $\neg B$(SR(e')), used to evaluate $\mathcal{I}(s)$, will return *True*, which means $s \in O$.

Cases o_2–o_3 are analogous to the previous one.

If s was added by o_4–o_6, then the proposition p at the left-hand side of the inference rule used to add s is a belief which either belongs to S or was added by Ri or o_1. In both cases, as the proof has shown so far, the BP will evaluate p as *True*. But for every possible consequence of o_4–o_6, it can be seen, case by case, that the truth of the premiss will force the BP to evaluate as *True* also the consequence. Hence, being s one of the consequences, it results $BP(s, S) = True$.

Finally, if s was added by one of the last two rules o_7, o_8, then the premiss which has caused its addition was a consequence of o_2 or o_3. In each of the two possible rules, the *absence* of the specific beliefs which have produced the addition of the default beliefs will force the BP to evaluate s as *True*.

Summarising, even when $s \notin Li$, s is evaluated *True* by the BP algorithm. Therefore, in both cases ($s \in Li$ and $s \notin Li$) it results $s \in O$, which implies $S^* \subseteq O$.

Part 2 ($O \subseteq S^*$): Let S be a subset of Li. Consider an element $\omega \in O$, where $O = \{\omega \in Lo \mid BP(\omega, S) = True\}$. We have to show that $\omega \in S^*$.

Let S' be the closure calculated by the BP through the inner rules which it adopts. Since these rules are exactly Ri, then $S' \subseteq S^*$, as $\mathbf{C}_{Ri}(S) \subseteq S^*$ because of property 3.4.

When the BP evaluates the expressions ω (and returns *True*), there are two possible cases: either *a*) ω is not modified by the syntactical transformations performed to calculate $\mathcal{I}(\omega)$ in step 1 or *b*) it is. Let us begin by considering the first case.

Part 2.a) If ω does not need to be simplified, then it must match one of the expressions evaluated directly by the *check*( ) function. In this situation, either *i)* the truth of ω is determined directly as $B(\omega)$, or *ii)* it is associated to a more

complex boolean function.

In case *i)*, since $BP(\omega, S) = True$, then $B(\omega) = True$, which means that $\omega \in S'$, and, therefore, $\omega \in S^*$, as $S' \subseteq S^*$.

In case *ii)*, ω can be one of the following expressions: 1) NOT(SR(e)), 2) SUND(e), 3) HUNK(e) or 4) NOT(x). If ω = NOT(SR(e)), then $\neg B$(SR(e)) must be *True*, from the definition of the function *check*( ). Hence, the proposition NOT(SR(e)) will be added to S^* because of o_1. Therefore, $\omega \in S^*$. Cases 2) and 3) are analogous, considering the rules of o_2–o_3. Case 4) requires a more careful analysis. If ω matches with NOT(x) (and ω is not of type $<A>$), then ω must be an *outer* expression which does not belong to the inner language. Therefore, the unification $\omega \cong$ NOT(x) will produce either x=HBEL(_), or x=HUND(_), or x=SUND(_) or x=HUNK(_). Since the truth value of ω (obtained as $\neg check(x)$) is *True*, then $check(x)$ must be *False*. Using this fact, and the *restrictions* which have been imposed on the model, for each of the previous cases it is possible to prove that $\omega \in S^*$. Let us examine in details only the first case, as the others can be proved in a similar manner.

Since the function $check(x) = check$(HBEL(e)) = B(HBEL(e)) returns *False*, then x=HBEL(e) $\notin S'$. Because of the restrictions imposed on the state, *at most one* of the two expressions HBEL(NOT(e)), HUND(e) can be present in S'. If one of them does belong to S', then one of the rules o_5 or o_6, respectively, will add NOT(HBEL(e)) to S^*, i.e. NOT(x) $\in S^*$. If *none* of them holds, then o_3 will add the default HUNK(e), which, in turn, will cause o_8 to append NOT(HBEL(e)). Therefore, in both cases, ω = NOT(x) $\in S^*$.

Summarizing, if ω is an expression which is not modified by the syntactical transformations of the BP algorithm, then $\omega \in S^*$.

Part2.b) Consider now the final possibility, in which ω is changed into ω' by one of the transformations applied by the BP algorithm[5].

Since $check(\omega') = True$, then, for what we have seen in the previous considerations, $\omega' \in S^*$. On the other hand, for each of the four possible ω' (namely, HUNK(e), SUND(e), HUND(e) and 'a'), either there is a rule in o_2–o_6 such that $\omega' \vdash \omega$, or ω' and ω must have been added *together*, as consequences of one of these rules. Therefore, also in this case, $\omega \in S^*$.

In conclusion, since every $\omega \in O$ also belongs to S^*, then $O \subseteq S^*$.

[5]Notice that if ω is a correct $<WFE1>$ expression, it cannot need more than *one* syntactical transformation.

Since $S^* \subseteq O$ (from Part 1) and $O \subseteq S^*$ (Part 2), then $S^* = O$.

□

6.3 Implementation

The actual implementation of the integrator system has been realized in 'C', a procedural language widely adopted for commercial applications and for which various compiling packages are available.

The software has been developed by taking into account two elements: 1) the planning system which will be used to solve the final standard problem , and 2) the specific characteristics of the application domain, namely, of the problem of *discourse planning.*

The planner adopted is 'IPP' [80], a system based on the Graphplan planning technique [15], which has been already introduced in Section 2.4. Apart from presenting very competitive performances, IPP allows the use of operators with conditional effects. This feature eliminates, in the process of integration, the need to convert the output of the last application of A1 into an equivalent problem which does not contain conditional operators.

On the other hand, one of the limits of IPP consists of the fact that it requires a *finite* planning language. This is due to the fact that every *variable* present in the given set of operators is automatically instantiated (in all the possible ways) at the beginning of the planning process.

This limitation constitutes an obstacle for the integration of the belief system developed, as the language $< WFF1 >$ contains an infinite number of expressions, due to the possibility of having SUP( ) expressions with an arbitrary level of nesting. The solution to this problem has been developed by adopting a 'lazy evaluation' approach, together with the exploitation of information derived from the specific domain problem definition.

In other words, the reduction of the general problem of planning to that of *discourse* planning allows to restrict the infinite sets of expressions generable by $< WFF1 >$ to finite collections containing only expressions belonging to what could be identified as the '*discourse context*'.

The discourse context can be intuitively defined as the set of concepts, extracted from the entire knowledge base of an agent, which are 'relevant' for the current

discourse, and which might need to be accessed during the process of construction of the discourse itself. In computer science terms, the discourse context can be thought to play the role of a 'buffer', or cache, between the discourse plan which is being built and the whole knowledge base.

The criterion used to select the concepts to include in this 'buffer' is based on the *relevance* of the concepts in relation to the contents of the discourse; the contents of the discourse are, in turn, determined by the communicative intentions, i.e., by the goals which need to be achieved by the communication process. According to the overview given in Section 4.1, we have assumed the initial state I, given to the planner, to have been identified using this criterion of selection, based on the the goal set G, which drives the plan construction process. In other words, I is supposed to contain *all* (and only) the relevant expressions which might be needed by the planning process itself to produce a complete discourse plan.

An example of algorithm for the selection of the relevant concepts to be included in I is described below, where G represents the set of (communicative) goals assigned to the specific discourse planning problem that the system is required to solve:

Init: if $g \in G$, the expression 'g' concerns the hearer's opinion towards a specific primitive event 'e' (e.g., if g = HUND(SR(E1)), then e=E1):

$$\forall g \in G, \text{ add the corresponding '}e\text{' and '}\neg e\text{' to } I$$

1. $\forall e \in I$, search the knowledge base for support relations such as SUP(e',e) or SUP(e',$\neg e$), and include all the relations and supporting events found — and their negations — into I[6];

2. apply step 1. recursively on the newly added events, until either *i*) real events, or *ii*) events with no support, or *iii*) events believed by both speaker and hearer are reached, and I will not be further extended.

The termination of this procedure is ultimately guaranteed by the fact that even the large knowledge base from which I is extracted must be finite, and thus contains a limited number of events.

After the execution of this algorithm, the set I can be assumed to represent the current *discourse context*, and the *domain* of each variable can be restricted to its

[6]Such events constitute the only ones which can be logically used to persuade the hearer about the validity or falsehood of 'e'.

intersection with I, which is a finite set. Therefore, the set of possible values of any variable of type 't' will simply consist of the (finite) collection of the expressions of type 't' which appear in I. The same principle of 'restriction' has been applied for the process of calculation of the actual closure I^* of the initial state I, which, otherwise, because of the presence of *default* rules in Ro, would have contained an infinite number of expressions (see Example 5.2, p. 128).

Finally, it should be pointed out that the integrator system implemented produces a grounded set of operators, by instantiating, initially, all the variables present in the set of schemes Op given in input. This strategy has been implemented for the following practical considerations:

- first of all, the instantiation of the variables of the operator schemes would have been automatically performed in any case by the planner IPP at the beginning of the planning process;
- secondly, in this way, the preprocessor assumes complete responsibility of the process of variable instantiation: consequently, the system will be able to properly control and optimize the process of *filtering* of the operator instances — through this process, the operator instances which are not useful for the specific discourse planning problem considered can be identified and eliminated in advance, before being considered for any plan;
- thirdly, dealing with grounded operators simplifies the implementation of the transformation algorithms A1 and A2. In fact, the presence of variables in the operator schemes would require the algorithms to consider every check for propositional matching (or inequality) as a condition of 'codesignation' (or 'non-codesignation') between variables and ground expressions of the language. Such conditions should then be attached to the operator's description.

Notice that the goal set G has actually been assumed to be already grounded.

6.3.1 The operators

In order to identify the 'tools' that the planner will be allowed to use for convincing the audience about the validity (or fallacy) of a specific argument, it is necessary to define the strategies of persuasion which can be employed in the discourse.

Because of the Sincerity Axioms (see page 7), neither rhetorical fallacy nor insincerity is admitted; hence, the basic mechanism of persuasion will have to rely

upon the *Modus Ponens* inference rule, also adopted by other discourse planning systems (see Section 2.4.2):

$$\text{HBEL}(a),\ \text{HBEL}(\text{SUP}(a,b)) \Rightarrow \text{HBEL}(b)$$

In other words, the system will assume the hearer will conclude b from a and $\text{SUP}(a,b)$. On the basis of this assumption, it is easy to deduce the 'Persuade' operator scheme, which was already introduced in Chapter 5 (Example 5.4):

Persuade(?a,?b)
Prec: HBEL(SUP(a,b)), HBEL(a), HUND(b),
SBEL(SUP(a,b)), SBEL(a), SBEL(b)
Add: HBEL(b)
Del: HUND(b)

Moreover, since the entire system of *inferential* beliefs of the hearer is expected to rely upon the same basic principle of *supporting event*, it is possible to develop three different techniques of belief *undermining*, based on the concepts of 'undercutting' and 'rebutting' defeater (see Section 2.3.2).

The first two techniques, belonging to the former category, consist of *spoiling* a belief which relies upon a specific supporting evidence. For example, suppose that HBEL(b) relies on the two beliefs HBEL(a) and HBEL(SUP(a,b)). If the hearer's confidence in either of the two beliefs a or SUP(a,b) is weakened, then also the confidence of the hearer in b will decrease, leading to an attitude of uncertainty (HUND(b)). If the belief in the argument a is weakened, we will talk of '*spoil-argument*' strategy, whereas the belief in SUP(a,b) will be undermined by a '*spoil-support*' strategy.

An example of combined application of these two techniques is shown in Figure 6.1. Suppose that, initially, the hearer believes the events E_1, E_2 and E_3, and also the two support relations SUP(E_2,E_1) and SUP(E_3, SUP(E_2,E_1)). Assuming the goal of the speaker to be HUND(E_1), one of the possible options would consist of undermining the *relation* of support of (the hearer's belief in) E_1, i.e., of adopting the *spoil-support* strategy, which achieves HUND(E_1) through the achievement of the *sub-goal* NOT(HBEL(SUP(E_2,E_1))). Such sub-goal, in turn, could be realized by employing the *spoil-argument* strategy, which would realize HUND(SUP(E_2,E_1)) through the achievement of a further sub-goal (namely, NOT(HBEL(E_3))), and so forth, until the newly generated sub-goal can be achieved either *i*) by simple establishment (i.e. it was already true in the initial state), or *ii*) through the '*attack*'

strategy (which will be explained in the following paragraphs) or *iii*) cannot be achieved at all, in which case the plan construction will have to backtrack.

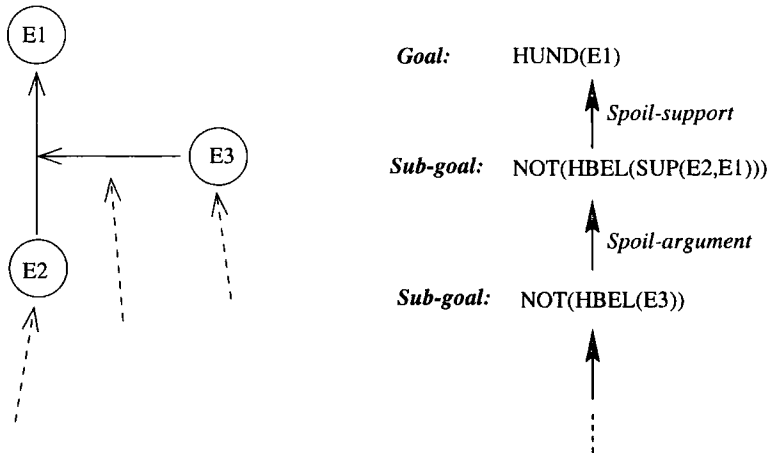


Figure 6.1: Spoiling the 'support' and the 'argument'.

The adoption of the previous two 'undercutting' techniques has led to the implementation of the 'Spoil-sup' and 'Spoil-arg' operators, which differ exclusively for the point of application of the process of sub-goaling: in the former, the SUP( ) event is targeted, whereas in the latter, the supporting argument is undermined. By way of example, below is reported the 'Spoil-arg' operator scheme, in which the precondition NOT(HBEL(*b*)) constitutes the sub-goal to be achieved in order to allow the execution of this operator:

Spoil-arg(?a,?b)
Prec: HBEL(a), HBEL(SUP(b,a)), NOT(HBEL(b)),
NOT(SBEL(a)), NOT(SBEL(b)),
NOT(HBEL(HR(a))), NOT(HBEL(SR(a)))
Add: HUND(a)
Del: HBEL(a)

The third undermining technique, mentioned above and equivalent to a 'rebuttal' of the argument, consist of *attacking* the belief in *b* by supplying the hearer with supporting evidence for the contrary event, NOT(*b*). The concept of 'attack' between two arguments has been also formalized by Dung in [32]. Presenting the hearer with evidence supporting the contrary of the current belief will lead to a 'balanced' situation (as that of Figure 4.3), in which the attitude of H towards the event *b* should be of undecision. The 'Attack' operator is shown below:

Attack(?a,?b)
Prec: HBEL(b), HBEL(a), HBEL(SUP(a,NOT(b))),
NOT(SBEL(b)), SBEL(a), SBEL(SUP(a,NOT(b))),
NOT(HBEL(HR(b))), NOT(HBEL(SR(b)))
Add: HUND(b)
Del: HBEL(NOT(b))

The complete list of the six operators developed for discourse planning is reported in Appendix A. In the actual specification of the operators, the variables '?a' and '?b' have been associated with specific types, so that they can be instantiated with any expression (which appears in I) admitted in such positions by the syntax of the language.

Let us dwell, for a moment, upon the characteristics of the operators introduced. First of all, as mentioned earlier, some of the terms of their precondition lists are supposed *not* to be true in the (*planning*) state which leads the operator itself to be chosen in order to realize a specific goal, but to be achieved as sub-goals by other operators. This happens, for example, for the precondition NOT(HBEL(E_3)) of the 'Spoil-arg' operator, as described in the example illustrated by Figure 6.1. Such 'preparatory' process, producing a *change* in the hearer's opinion towards part of the supporting evidence, gives the operator the necessary *persuasive force* to succeed and achieve the expected perlocutionary effect.

However, this behaviour cannot be explicitly imposed to the planning machinery without the use of a *filter* list and of an 'abstract' operator, encapsulating a body with sub-goal list. Alternatively, a further operator could be defined, in which the spoiling is 'prepared'. This new operator, having HBEL(b) as precondition, would simply have to enable the subsequent application of 'Spoil-arg' through the addition of a new proposition 'flag' to the state. Such 'flag' should then constitute one of the preconditions of 'Spoil-arg'. Nevertheless, this would require the introduction, in the language of beliefs, of an unnecessary new term completely 'extraneous' to the syntax defined. Considering the demonstrative aim of this implementation, as a first approximation, the operators 'Spoil-arg' and the analogous 'Spoil-sup' have been adopted simply as specified above. As it will be shown, this does not prevent the system from generating valid examples of persuasive discourse plans.

Another interesting feature of the operators presented consists of the fact that their precondition lists contain conditions which specify the state of the *speaker* with respect to the beliefs which are involved in the scheme. Such conditions force

the operator to respect the requirement of *sincerity* of the speaker (Sincerity Axiom I).

Moreover, it should be noticed that the 'Spoil' and 'Attack' operator schemes are applicable if and only if the belief which the speaker is trying to spoil (or attack) does not constitute, in the hearer's view, a *real* experience. In other words, an event derived from the sensorial experience of an agent is *never* spoilable. This condition is guaranteed by the presence of the term NOT(HBEL(HR(a))) in the precondition lists of the operators. Notice that the precondition NOT(HBEL(SR(a))) in the 'Spoil' and 'Attack' schemes would not be strictly required, if not to avoid the possibility of contradiction. In fact, if such precondition were not present, the state could end up containing both the events HUND(a) and HBEL(SR(a)); but the latter belief would produce, through the inference rule i_4, the addition of HBEL(a), which would be in conflict with the former. This would seem to imply that, once the hearer has come to believe that an event constitutes a real experience for the speaker, such conviction will never change, even if mistaken. Nevertheless, this is not so, as it is always possible to introduce a further operator in which the belief HBEL(SR(a)) is required in the preconditions and then explicitly deleted.

Notice also that the operators implemented satisfy the condition of *coherence*, i.e., if the state to which they are applied is coherent (that is, it does not violate the *restrictions* of mutual exclusivity imposed on the model), then also the resulting state will result to be coherent.

Finally, the requirement of sincerity of the speaker during all the process of persuasion leads to an interesting property of the model, namely, the '*monotonicity*' of the discourse. This means that during the same discourse plan, the speaker will never try to convince the hearer about the validity of two *opposite* events, e and NOT(e). In fact, in this implementation, the coherence of the state implies that explicitly contradictory events cannot be simultaneously believed; moreover, the speaker's attitudes towards the 'primitive' events do not change during the same discourse. Hence, on the basis of the first sincerity axiom, the speaker will not be allowed to try to convince the hearer of the validity of what (s)he does not believe to be true, that is, of two explicitly contradictory events E and NOT(E).

The premisses upon which a persuasive discourse plan will rely are represented by the *real experience* events. These constitute the common 'ground' which need to be shared by the speaker and the hearer before any attempt of persuasion is carried out. A simple and direct way in which the speaker can make the hearer aware of his/her real experiences consists of *telling* the audience about them. Since the

audience is expected to believe in the speaker's sincerity (second Sincerity Axiom), asserting the fact that '*e*' constitutes a real experience will force the hearer to accept such fact (and all the direct consequences). The specific speech-act operator which realizes this is the 'Assert-real' operator scheme, reported below:

Assert-real(?a)
Prec: SR(a), HUNK(SR(a))
Add: HBEL(SR(a))
Del:

Notice the empty delete list: since the proposition HBEL(SR(a)) is added and nothing is deleted, in order to guarantee the maintenance of coherence, the operator must verify that the default belief HUNK(SR(a)) hold at the moment of its application. A further operator, with a different precondition — e.g., HUND(SR(a)) — could be introduced to allow the application of the same speech-act under different belief circumstances.

An operator similar to 'Assert-real' has been adopted for the introduction, in the discourse, of an event (not necessarily a real experience) which has never been conceived before by the hearer (i.e., HUNK(a)). The expected resulting attitude of the hearer towards a newly introduced concept is that of uncertainty (see operator 'Assert' in Appendix A).

With the adoption of these quite simple operators it is already possible to show some interesting discourse plan construction. However, before moving on to the description of some examples, there are two final points which should be made, one concerning practical aspects of the implementation, the other concerning a more 'theoretical' property.

First of all, consider the process of transformation of the BP planning problem as described by the formula 6.1 of Section 6.2. In the actual implementation, the five inference rules i_1–i_5 have been divided into the two acyclic blocks $Ri_1 = \{i_1, \ldots, i_4\}$ and $Ri_2 = \{i_5\}$. These two blocks are integrated in the operators through algorithm A1, during the first two steps of the process. Algorithm A1 is expected to generate new sets of *conditional* operators; however, because of the *specific characteristics* of the operators and of the inference rules adopted, the sets of transformed operators resulting from the two initial steps will not contain *any* conditional effect.

The reasons of this result lie in the definition of the algorithm A1: the addition of conditional effects during the transformation of an operator is due to two possible cases: 1) a proposition might not be deleted if it is a *consequence* of one of the

inference rules; 2) the deletion of a proposition which is a *premiss* of one of the rules might lead to the deletion of other propositions. The second condition never occurs: *none* of the delete lists of the operators matches with any of the premisses of the rules. The first condition can occur only in rules i_4 and i_5. For example, in 'Spoil-arg', the proposition HBEL(b), which matches the consequences of both rules, is deleted. However, the conditions under which this deletion would be restricted (i.e., NOT(HBEL(SR(b))) for i_4 and NOT(HBEL(HR(b))) for i_5) are already required by the *main* preconditions of the operator. Hence, the produced conditional effect can simply be 'inglobated' in the main effect of the operator. The same happens for the other operators, 'Spoil-sup' and 'Attack'.

Therefore, in the actual implementation of the process of transformation, A1 did not have to be extended in order to accept also conditional operators. Notice that since the planner IPP does actually allow the use of operators with conditional effects, the final application of A1 is allowed to produce output containing conditional operators. However, empirical results have also shown that if the algorithm A1 is modified so that it calculates the new add list A' not simply as $\mathbf{C}_R(A)$ but as $A' = \mathbf{C}_R(A) \setminus \mathbf{C}_R(P)$, then *all* of the conditional effects introduced by the last step of the transformation disappear.

To sum up, the absence of conditional effects in the transformed operators seems to be linked with the property of *coherence* that the operators have to satisfy. In fact, this was the reason for the introduction of the precondition NOT(HBEL(SR(b))) in the 'spoil' and 'attack' operators. Although the symmetrical precondition NOT(HBEL(HR(b))) has been added for the specific 'semantics' adopted by the model, its presence can also be justified on the same basis: the need to avoid contradiction and to maintain coherence.

The second interesting point is represented by the following claim, which, somehow, justifies the adoption of the more simple and restricted syntax $<WFE1>$ (which allows only one level of belief nesting) for the implementation of the discourse planning system:

> **Claim (Hearer Model)** *If the two Sincerity Axioms hold, planning successful communication does not require the speaker to maintain a representation of the hearer's model of the speaker's beliefs.*

The previous claim can be justified through the following argument: the necessity of representing the hearer model of the speaker beliefs (using a second level of nested beliefs) arises iff the hearer's model of the speaker differs from the real

speaker's set of opinions. In fact, in absence of differences, the hearer's model simply coincides with the actual speaker's state of beliefs, and does not need to be explicitly represented. There are two symmetrical situations which require to model a difference between the hearer's opinion about the speaker's attitude towards an event and the speaker's own attitude towards the same event:

1. *Deception introduction*: the speaker intends to tell the hearer a lie;
2. *Deception elimination*: the speaker thinks that a 'misunderstanding' - or *lack* of information - in the hearer's beliefs is already present, and wants to rectify it.

Under the assumption that the speaker is sincere (Axiom I), the first situation shall never occur. With regards to the second case, in which an initial difference is given, it is interesting to show that the speaker is not required to model the situation in order to solve it, but needs only to assume the two sincerity axioms to hold. In fact, let's suppose the speaker's belief about the event E to be different from the hearer's model of it (e.g. SUND(E), HBEL(SBEL(E))). There are two possible cases:

a) the event E is relevant for the current discourse (i.e. it is going to be used in the discourse which the speaker is planning);

b) the event E is not relevant for the current discourse, and will not be mentioned.

In the first case, building a discourse plan which makes use of the event while ignoring the initially given misbelief might lead to think that part of (or the whole) plan will fail, as based on wrong premisses. Nevertheless, when the speaker actually mentions E in the discourse, because of the first sincerity axiom, (s)he can only talk about E by asserting what is his/her real opinion about it (in the example, the speaker might try to achieve HUND(E), by asserting that (s)he, too, is undecided about E). In force of the second axiom, the hearer must believe the speaker to be honest and truthful: the immediate consequence of the assertion, then, is that the hearer will actually realize the misunderstanding, and correct the mistaken (or *uncertain*) opinion[7]. Hence, in this case, the modelisation of the 'wrong' belief is not necessary: ignoring the situation does not affect the successful completion of the

[7]In a real dialogue, this would probably involve a short interruption, in which the hearer would ask the speaker to confirm once more the assertion, before finally rectifying the belief.

plan, as the misunderstanding is automatically rectified as soon as the concerned event is mentioned in the discourse.

In the second case, any misunderstanding about the event E will not affect the outcome of the speech, as E (and the speaker's opinion about E) will not be used in the discourse. Thus, even in this case, the speaker can ignore the initial misbelief, leaving its resolution to a future occasion in which it will be solved as in case a).

6.4 Examples

The discourse planning system implemented has been applied to twelve different examples, all of which adopted the same basic set of operators — introduced in the previous section — but different initial belief states, and to achieve different goals. For each example, the discourse plan has been produced by first applying the integrator to the initial state and operators, producing a new problem suitable for the IPP planner, and then running IPP on the planning problem obtained. The CPU execution time (on a SPARC workstation) for the transformation process varies from about ten seconds to a maximum of six minutes in the most complex example; the planning time varies from a minimum of 0.25 seconds to 22 seconds. Nevertheless, it should be noticed that the code executed to perform the transformation is far from being optimised for speed: the program represents only a 'prototype' system which has been developed for experimental and demonstrative purposes, rather than for competitive performances.

Of the twelve examples, only the most representative have been reported here. Before illustrating the examples which have been chosen, though, it is necessary to introduce the graphical notation adopted to represent the events contained in the initial state. Figure 6.2 contains the five basic 'units' used to represent the events which are objects of various attitudes of the speaker and hearer. Each basic representation has also been described by a proposition, which appears besides the corresponding graphical model.

Structure (a) represents the basic relation of support between two events. A bold circle containing an event, as illustrated in case (b), qualifies the event itself as a *real experience*. Notice that the graphical representation does not clarify whether the event is 'real' from the speaker or from the hearer point of view. Case (c) shows how a bold 'arrow' is adopted to indicate that the support relation between two events is based on a real experience (i.e. on *induction*). Also in this case, the induction could be part of either the speaker's or the hearer's experience. Notice

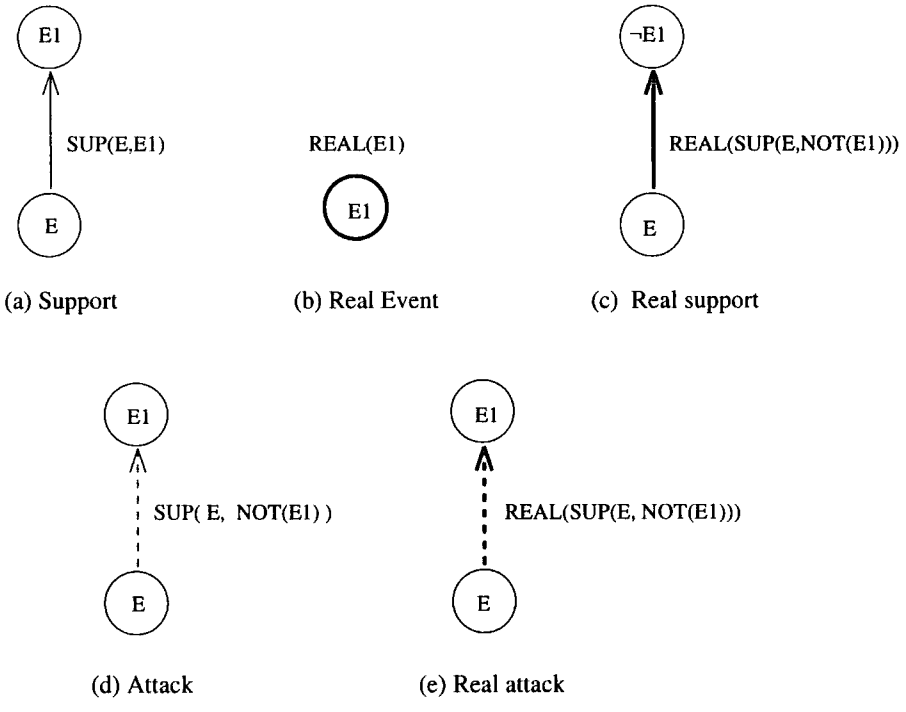


Figure 6.2: Graphical notation of beliefs justification and grounding.

also the use of the abbreviation '¬ E1' for the expression NOT(E1).

The dashed arrows in (d) and (e) have been adopted to underline the relation of *attack* between two events; notice that these two graphical symbols would not be strictly necessary, as already representable through cases (a)–(c). In fact, as it appears from the corresponding propositional forms, cases (c) and (e) represent the same situation. Summarizing, 'normal' events are represented with circles, support events with arrows, and 'attack' relations with dashed arrows; any of these can be qualified as 'real' adopting a bold graphic.

Finally, it should be pointed out that since support relations are events themselves, the graphical representation also allows 'arrows' in each of the positions occupied by 'circles' in Figure 6.2.

The first, simple example examined, showed in Figure 6.3(a), is a variation of an example taken from Young and Moore [153], who adopted it in order to illustrate a discourse plan analogous to that reported below, although using a more convoluted graphical representation. In the figure, some of the attitudes of the speaker and hearer have been added besides the events, in order to clarify the situation. For example, the event E1 is a *Speaker*'s real experience; also, the hearer believes (HBEL) the support relations between E3 and the other two events, but

is undecided about the validity of E3. Notice that where not otherwise specified, the hearer is assumed — by *default* — to 'unknow' the event (e.g., HUNK(E2), HUNK(E1), HUNK(SR(E1)), etc.).

The set of propositions constituting the initial state given in input to the integrator system (corresponding to the graphical representation of Figure 6.3(a)) is reported below:

I={ E3, E2, SR(E1), HUND(E3), SUP(E3,E2), SUP(E2,E3), SUP(E3,E1), SUP(E1,E3), SUP(E1,E2), HUND(SUP(E1,E2)), HBEL(SUP(E3,E2)), HBEL(SUP(E2,E3)), HBEL(SUP(E3,E1)), HBEL(SUP(E1,E3)) }

The goals to be achieved are { HBEL(E1), HBEL(E2), HBEL(E3) }. It is important to point out the 'circularity' of this example: first of all, the two pairs of events (E1,E3) and (E2,E3) have circular supporting relations — i.e., each event of a pair is supported by and supports the other one. This situation is acceptable only because one of the events (namely, E1) is a real experience, and can be taken to be the 'basis' of such circular local structure of beliefs of the speaker.

Secondly, since also the hearer believes those support relations, in order to plan a persuasive discourse to convince H about the validity of all of the three events, it will be sufficient to achieve such effect for one — *any* — of them, and the other two will follow through the application of the '*Modus Ponens*' (Persuade) operator. In this specific case, the initial basis upon which the rest of the persuasion will rely consists of the (speaker's) real experience E1, which will be 'shared' with the hearer at the beginning of the discourse.

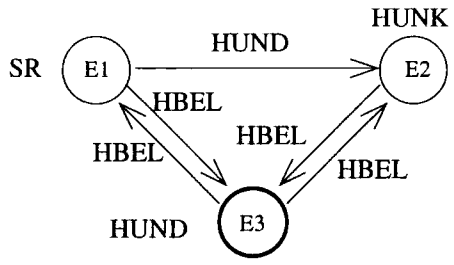
Indeed, after the transformed problem is given in input to IPP, the results produced are as follows:

```
ipp:  found plan as follows

time step 0:  assert-e2
              assert-real-e1
time step 1:  persuade-e1-e3
time step 2:  persuade-e3-e2

ipp:  used 0.36 seconds total time
```

The simple discourse plan found consists of four 'steps', but, as the result underlines, the first two can be applied 'simultaneously', as they do not *interfere* during



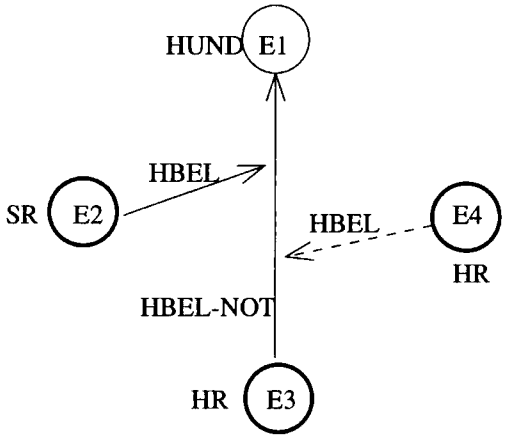
E1 = "Lucentio has asked for Bianca's hand in marriage"

E2 = "Bianca has always been Lucentio's preferred model for painting"

E3 = "He considers her the fairest of Signor Baptista's daughters"

Goals: { HBEL(E1), HBEL(E2), HBEL(E3) }

(a)



E1 = "H will miss the 8am train to London"

E2 = "There is a traffic jam on the way to the station"

E4 = "H can drive very fast"

E3 = "H is late for the 8am train"

Goals: { HBEL(E1) }

(b)

Figure 6.3: Graphical representation of two initial states.

plan execution. This is why IPP indicates the plan as composed of only three steps. In other words, the order of application of the first two operators during the actual plan execution is not relevant for the achievement of the goals.

The plan produced consists of introducing the hearer to the event E2, which was previously *unknown*, and asserting that the event E1 is a real experience. After this premiss, the speaker will use the 'modus ponens' strategy on E1, exploiting the assumption that the hearer believes the support relations going from E1 to E3, and from E3 to E2. Notice that the possibility of convincing the hearer about E2 using E1 and the relation SUP(E1,E2) is 'prevented' by the fact that the speaker does not have supporting evidence to persuade the hearer about SUP(E1,E2). Nevertheless, the introduction of another operator of persuasion that assumes the hearer to believe the events for which no support can be provided would make possible the generation of such a plan.

A possible natural language interpretation of this first simple plan could be as follows:

assert-e2	"Bianca has always been Lucentio's preferred model for painting.
assert-real-e1	**As a matter of fact, I was present when** he asked for her hand in marriage:
persuade-e1-e3	**this clearly implies that** he considers Bianca the fairest of Signor Baptista's daughters.
persuade-e3-e2	**Hence, no wonder that** she has always been his preferred model for painting."

The 'cue words' added to the structure have been outlined in bold; apart from such parts, which can be deduced directly from the meaning of the operators applied, the discourse has been obtained simply by substituting the semantic material to the corresponding primitive events E1,E2 and E3.

In the second example, showed in part (b) of Figure 6.3, the support relation SUP(E3,E1) is *attacked* by the event E4 (which is a hearer's real experience), and supported by event E2 (which is a speaker's real experience of which the hearer is not aware). Because of the attack, which relies on the hearer's ability to drive fast, the hearer (H) believes that NOT(SUP(E3,E1)), i.e. that the fact that (s)he is late does not imply that (s)he will miss the train. The plan produced to achieve HBEL(E1) is reported below:

```
ipp:  found plan as follows

time step 0:  assert-real-e2
time step 1:  attack-e2-not-sup-e3-e1
time step 2:  persuade-e2-sup-e3-e1
time step 3:  persuade-e3-e1

ipp:  used 0.42 seconds total time
```

In order to convince H that E1 is true (i.e., that (s)he will actually miss the train) the speaker asserts that E2 is real, and that this undermines H's believe in NOT(SUP(E3,E1)), since SUP(E2, SUP(E3,E1)) [8]. As a result of the attack, the H is expected to assume an attitude of undecision towards the support event. After that, the speaker uses E2 again, this time to persuade the hearer that not only was the belief in NOT(SUP(E3,E1)) wrong, but, also, that SUP(E3,E1) is

[8]Notice that the formula 'attack-e2-not-sup-e3-e1' should be interpreted as 'E2 attacks NOT(SUP(E3,E1))'.

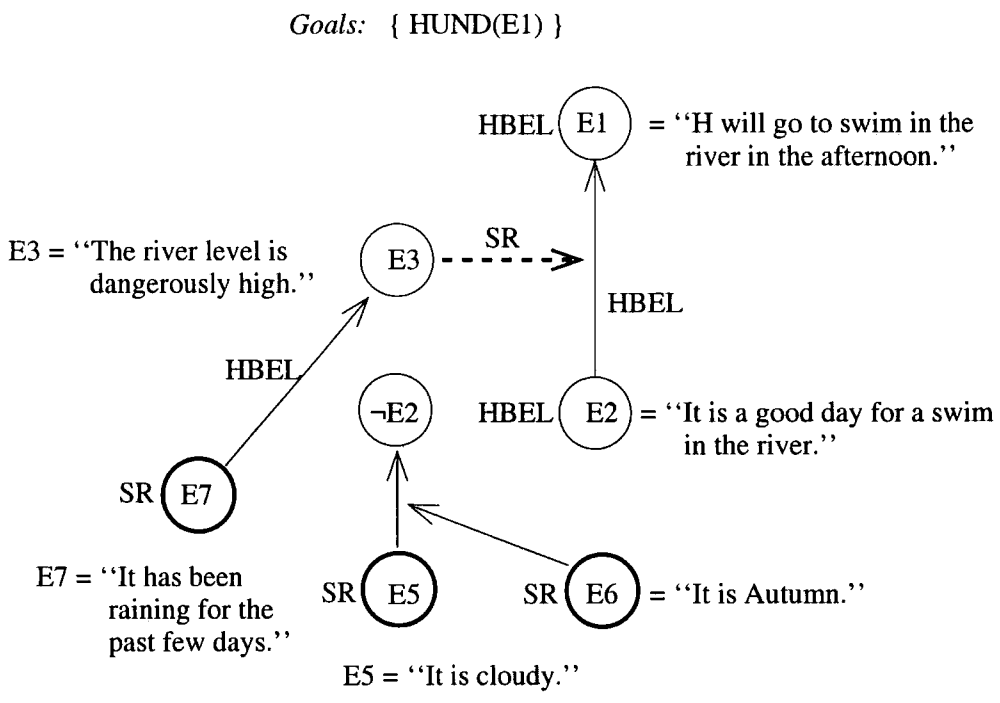


Figure 6.4: Planning problem leading to 'spoil' plan.

actually true. This can be done because H is expected to believe that SUP(E2, SUP(E3,E1)). Finally, using the support now believed by both, the speaker concludes that H should also believe E1.

Notice that in this example, the plan consists of four steps which are not executable in parallel, as the consequence of each operator is a necessary condition for the application of the subsequent one.

Having examined the functioning of the 'Persuade' and 'Attack' operators, the following example has been chosen to illustrate how the 'Spoil' scheme can be applied. The initial situation and corresponding goal are shown in Figure 6.4.

Essentially, the discourse plan produced for this case consists of convincing the hearer that the support relation SUP(E2,E1) is not valid, and, therefore, the belief in E1, based upon such support, should be abandoned. This is exactly what happens in the last two steps of the plan found by IPP:

```
time step 0:  assert-real-suprnotr-sup-e3-e2-e1
              assert-real-e7
              assert-e3
time step 1:  persuade-e7-e3
time step 2:  attack-e3-sup-e2-e1
time step 3:  spoil-sup-e1-e2

ipp:  used 3.36 seconds total time
```

The 'spoil-sup' operator is based on the state of undecision produced in the hearer by the previous attack on SUP(E2,E1) coming from E3. Notice, in step 0, the expression 'suprnotr-sup-e3-e2-e1', which corresponds to the 'linearisation' of the predicative form SUP(E3, NOT(SUP(E2,E1))), representable with a binary-tree structure. The linearised form is obtained by 'visiting' the tree according to a 'node-left-right' order of precedence, and then by *shifting* all the primitive events to the ending part of the expression [9].

It is interesting to note that, for this example, there exists a second valid plan, which achieves the spoiling of E1 through the attack of E2 (the supporting *argument*) instead of SUP(E2,E1) (the *support* relation). This alternative plan would consist of *i*) persuading H that SUP(E5,NOT(E2)); *ii*) attacking E2 from E5; and *iii*) spoiling E1 on the basis of lack of confidence in the argument E2 ('spoil-arg' instead of 'spoil-sup'). The final plan, however, would contain the same number of steps of the previous one.

A richer example, in which the 'spoil-arg' operator is actually chosen, is given in Figure 6.5. Also in this case, the goal consists of 'shaking' the hearer's belief in an event (E1) by undermining its basis.

[9]The linearisation also attaches the 'NOT' appearing inside a support directly to the SUP predicate in form of a desinence. E.g., 'supnotr' is obtained from SUP(_ , NOT(_)), the character 'r' indicating that the right branch should be negated. Similarly, 'supr' means that the right branch of the SUP will consists of a further support event.

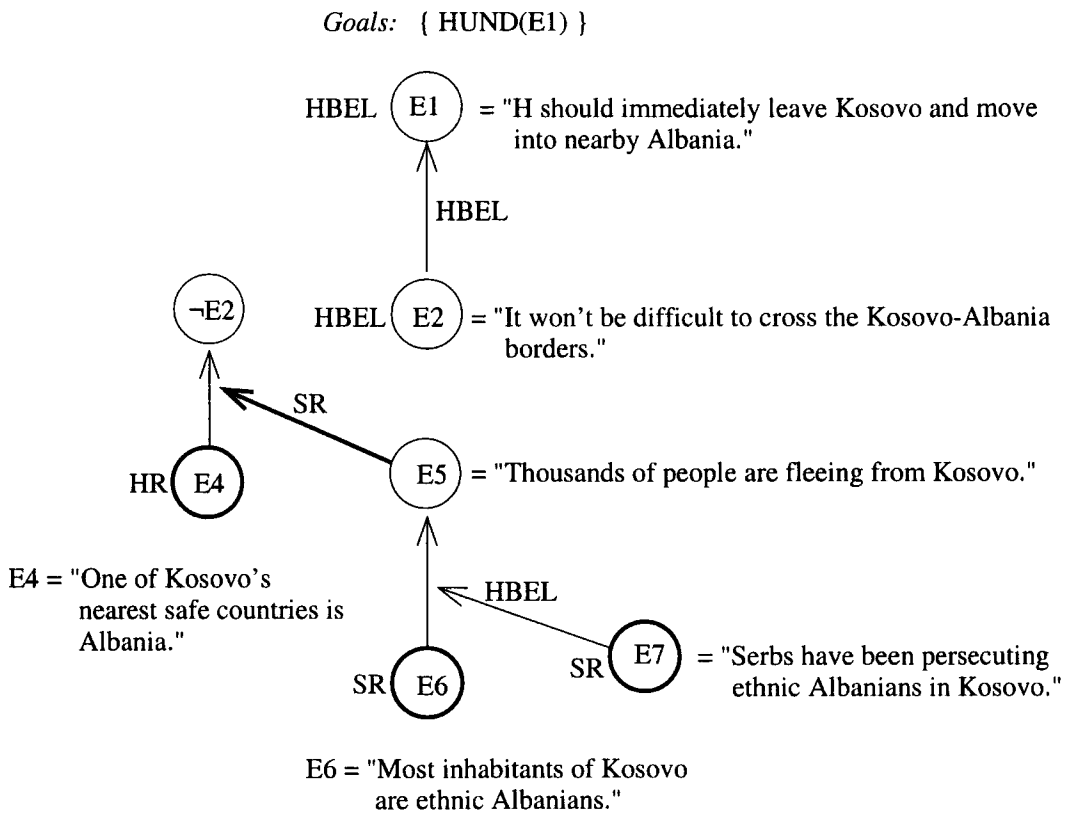


Figure 6.5: Situation leading to a 'spoil-arg' plan.

The plan found in this case is the only one possible; it proceeds, as usual, from the 'basis' of the structure (asserting real events, i.e., the premisses of the discourse), and then persuading the hearer of a support event — SUP(E4, NOT(E2)) — which leads to the final attack of E2, with consequent spoiling of E1:

```
time step 0:  assert-real-supr-supnotr-e5-e4-e2
              assert-e5
              assert-real-e6
              assert-real-e7
              assert-sup-e6-e5
              assert-supnotr-e4-e2
time step 1:  persuade-e7-sup-e6-e5
time step 2:  persuade-e6-e5
time step 3:  persuade-e5-supnotr-e4-e2
time step 4:  attack-e4-e2
time step 5:  spoil-arg-e1-e2

ipp:  used 2.98 seconds total time
```

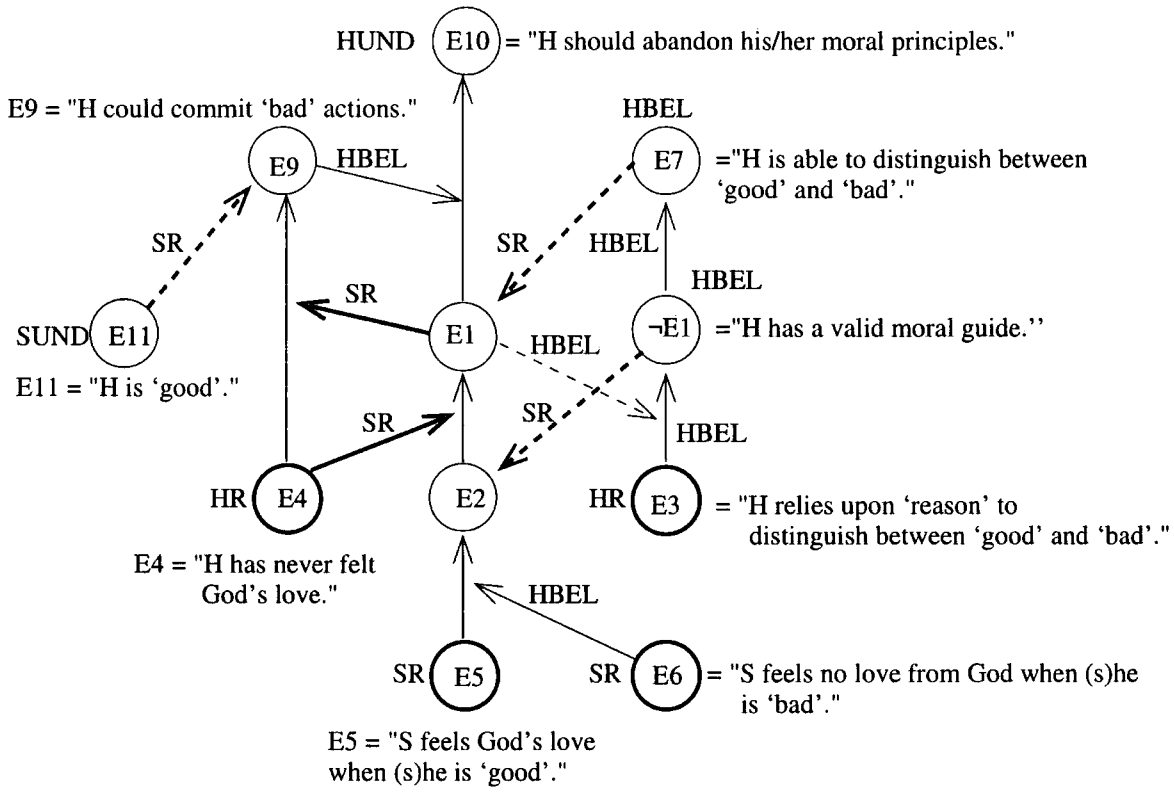


Figure 6.6: A more complex discourse planning problem.

The example shown in Figure 6.6 completes this section. This example differs from the others in the sense that it contains many more events, part of which are not actually necessary for the construction of the plan. The presence of more elements notably increases the number of grounded operators to be processed, and makes the search space bigger, producing a longer time of problem transformation (around six minutes) and planning.

It is interesting to notice that the final plan, which achieves both of the two goals HBEL(E10) and HUND(E7), consists of an initial part — up to step 2 — aimed at convincing H about the lower and central 'area' of the structure, and of two subsequent parts, based on the previous, in which first the right and then the left-side branches are used to produce the expected goals.

The plan produced by IPP for this example is reported below.

```
time step 0:  assert-e9
              assert-real-supr-sup-e1-e4-e9
              assert-e2
              assert-real-e5
              assert-real-e6
              assert-sup-e5-e2
              assert-real-supr-sup-e4-e2-e1
              assert-sup-e2-e1
              assert-sup-e4-e9
              assert-sup-e1-e10
time step 1:  persuade-e4-sup-e2-e1
              persuade-e6-sup-e5-e2
time step 2:  persuade-e5-e2
time step 3:  attack-e2-not-e1
time step 4:  spoil-arg-e7-not-e1
              persuade-e2-e1
time step 5:  persuade-e1-sup-e4-e9
time step 6:  persuade-e4-e9
time step 7:  persuade-e9-sup-e1-e10
time step 8:  persuade-e1-e10

ipp:  used 22.34 seconds total time
```

In contrast with all of the previous examples, this one contains also events and relations which are not strictly necessary to identify and build the discourse plan that achieves the specific communicative goals. However, since the initial state I is regarded as the equivalent of the current 'discourse context' (see Section 6.3) and is supposed to have been 'extracted' from an underlying, larger knowledge (or belief) base, such unnecessary elements are most likely to be included in it, because of their alleged *relevance*, which could have been assumed on the basis of their connection with the two assigned goals.

Chapter 7

Evaluation

This chapter performs the evaluation of the results of this thesis according to the criteria specified in Section 2.5. Each of the three parameters adopted — expressiveness, efficiency and correctness — is considered in respect to the two different aspects of the results of this work, that are *i*) the discourse planning system obtained from the integration of a belief system with a planner and *ii*) the belief system evaluated as a stand-alone module.

7.1 Expressiveness for Discourse Planning

The first point required by the expressiveness of the discourse planning system consists of its ability to represent the *intentional* structure of the discourse which is being constructed. The model developed adopts the planning technique as the basis for such representation, where the communicative goals which lie behind the formation of the discourse segments are represented by the goals which the operators achieve.

The operators schemes specifically adopted in the system (listed in Appendix A) can be divided into two different categories, namely, *abstract* operators and *primitive* operators. To the first class belong the 'Attack', 'Persuade' and 'Spoil' operators, whereas 'Assert' and 'Assert-real' constitute the second group. The abstract operators are aimed at achieving *perlocutionary* goals, whereas primitive operators can be seen as representing *locutionary* speech acts[1].

[1]The *illocutionary* effects, as defined by Searle [128], consist of the hearer *understanding* the content of the message, and belong to a lower level of plan refinement. In fact, they involve the choice of the specific 'surface features', such as the terminology adopted and the linguistic register, which are necessary for the actual realisation in NL of the message. However, as declared

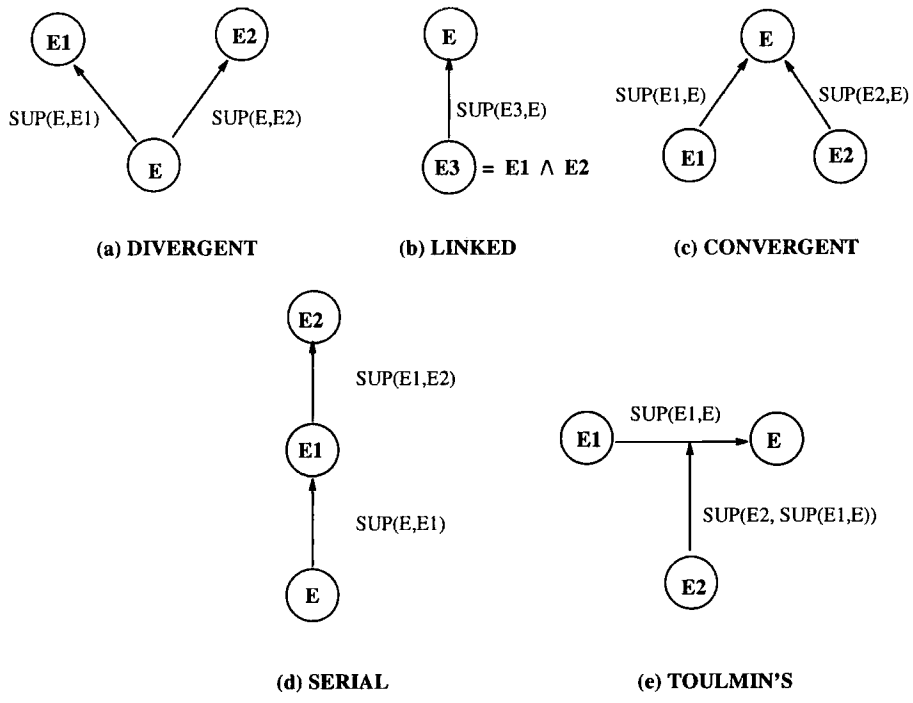


Figure 7.1: The five basic argument structures realized with '*sup*'.

For each operator, the communicative goal is identified by the (single) proposition which appears in the *Add* list. For example, in case of the 'Persuade' operator, the communicative goal is 'HBEL(p)', where 'p' is a specific proposition. Hence, the intentional structure of the discourse is captured by the plan which is being constructed, which contains all the elements necessary to derive the relations between the various operators, and, thus, between the different intentions which generate the different discourse segments. Such information is of enormous importance in case of a replanning due to a partial plan failure.

The second characteristic required to the model consists of its ability to represent the *functional* relations which exist between different parts of a persuasive discourse as accounted by the *standard* analysis and by the Toulmin schema (see Figures 2.2 and 2.3).

Figure 7.1 shows how each of these relations can be expressed through 'support' events, which are elements of the language adopted in the model. It should be noticed that the linked structure (Fig. 7.1(b)) has been realized by assuming an event, E_3, to be equivalent to the *conjunction* of the two events $E_1 \wedge E_2$.

If the two events E_1 and E_2 which constitute E_3 need to be considered sepa-

at the beginning, these aspects have not been treated in this work.

rately, because, for example, they have separate supports, then it is possible to add, beneath E_3, a Toulmin-like structure, in which E_1 and E_2 are *interchangeable* premisses which support the event conjunction $E_3 \equiv E_1 \wedge E_2$. Such a situation can be formalised by the logical expression

$$E_2 \rightarrow (E_1 \rightarrow (E_2 \wedge E_1))$$

in which the order of the two premisses E_2 and E_1 can be inverted without losing the validity of the formula.

The third and fourth features required to the expressiveness of the model consist of the ability to represent, for the speaker and for the hearer, support and attack relations, the latter including rebutting and undercutting defeaters.

A support relation is obviously represented by a support event SUP(E,E′). A rebutting attack can be represented simply by SUP(E, NOT(E′)), that is, by a support for the negated event. An undercutting attack consists of an event which attacks (directly, i.e. as a rebuttal) the relation of support between two events. This can be formalised in terms of support events as

$$\text{SUP(E, NOT(SUP(E',E''))}$$

The subjective, non-strictly logical character of such relations results from the fact that there are *no* specific requirements on the contents of the events E, E′ and E″ in any of the previous expressions. This means that the speaker could believe that a certain event constitutes a support for another event, even though such support is not logically valid.

In order to separate the hearer model from the speaker's attitudes, it is sufficient to place the prefix 'HBEL' in front of the specific event, support or attack relation. Notice that in the more complex $<WFE>$ syntax, the model of the hearer can, in turn, embed a (hypothetical) model of the speaker, allowing expressions with two levels of belief nesting such as HBEL(SBEL(...)).

7.2 Expressiveness of the Belief System

The first requirement of expressiveness to be satisfied by the belief system consists of allowing the model to deduce new beliefs from others using specific rules of inference. Ample evidence of this ability has been given in Chapters 3, 4 and 6, where the paradigm for belief systems has been, respectively, defined, implemented

and integrated.

Section 3.1 clearly pointed out the similarities between the model and the *formal theories* in logic. A possible objection to this analogy consists of the fact that the *modus ponens* inference rule, adopted by most of the formal logics, contains *two* premisses, whereas the formal correctness of the process of integration described in Chapter 6 is based on the assumption that the inference rules pre-processed contain only a single premiss (or, alternatively, the negation of a disjunction of premisses, for default rules). Hence, the classical propositional logics could not have been integrated in any planner; not, at least, using the specified method.

Nevertheless, the requirements imposed on the inference rules adopted by the belief system have been introduced explicitly to allow the system to be *integrated* into a planning framework. In such framework, it is possible to specify *operators* which can easily replace the *modus ponens* rule, or even other rules of inference. For example, below is presented a primitive version of a *modus ponens* operator[2]:

Preconditions:	SBEL(a), SBEL(SUP(a, a'))
Add:	SBEL(a')
Delete:	SBEL(NOT(a'))

Notice that this operator could be correctly applied also to the situation where SUND(a') holds, for such belief would not be explicitly represented in the state, and would not need to be deleted, either.

The adoption of this kind of operators allows the model to perform inference and to represent it *dynamically*, that is, without the need to explicitly add to the current set of beliefs the conclusions which the application of the inference rules has led. Therefore, this strategy represents also a possible way of approaching to the problem of logical omniscience: in this perspective, the beliefs of the agent are not closed under logical consequence, as only the deductions which are useful for the plan are considered. In other words, the system is supplied with the *tools* to deduce, potentially, any possible logical consequence, but the actual deduction of new beliefs depends entirely on the development of the plan, which is driven by the specific *goals* and, possibly, subject to time or space constraints.

The second parameter specified by the criteria for the evaluation of the expres-

[2]An analogous version of this operator has been presented by Reed *et al.* in [123] for the hearer model of beliefs.

siveness of the belief system consists of the ability to tolerate contradiction and uncertain information.

If two opposite events, E_1 and NOT(E_1), are believed, the contradiction will not necessary spread through the rest of the system, unless the logical formula $(p \wedge \neg p \rightarrow q)$ (*ex falso quodlibet*) is explicitly adopted by the system through an 'inference' operator which models such rule of deduction. Although two contradictory events can lead to contradictory conclusions, the propagation is limited to the *contexts* in which such events have a *relevance*, and restricted to situations where these events can be used as premisses for others.

In presence of contradictory information, however, the model allows also to *suspend* the judgement, and simply adopt the attitude of *undecision* towards the two conflictual beliefs. For example, if E_1 and NOT(E_1) both have supporting evidence, the expected attitude of a rational agent will be that of temporary suspending the judgement until further decisive evidence has been brought to light; in the model, this is realized by adopting the attitude SUND( ) towards both of the contradictory events.

This also illustrates how the system can deal with uncertainty. The attitude of undecision is normally adopted also in case of events for which no evidence at all has been collected yet, neither positive nor negative. This occurs, for example, when a new event or concept is simply *formulated* in an agent's mind, originated by an external input or by an internal 'intuition'.

Finally, the model allows a subjective and context-dependent structure of belief justification and grounding. In fact, no specific requirements have been imposed on the level of regress of justification. Such level can be decided subjectively, according to the context or to the specific characteristics of the audience addressed. The representation allows an event to be believed without any support, or to be accepted only if the hearer (hypothetically) does so, or to be believed on the basis of *real* (sensorial, as required by Armstrong) experiences.

The flexibility of the model enables the definition of a belief system based on the coherence theory, on the foundations theory, or on both theories; for example, the model can adopt a foundations approach whenever it is believed to be appropriate for the specific *context*, and be generally based on a coherence view, in which a belief does not need further support if it is not (supposedly) questioned by the audience. This would be in line with the idea of *locality* and 'isolated' grounding of beliefs, put forward by Doyle in [31] and discussed in the last part of Section 2.2.2.

7.3 Efficiency of Discourse Planning

The efficiency of the belief system considered as a module to be integrated into a planning system to produce persuasive discourse plans is evaluated according to the two parameters defined in Section 2.5, namely, the efficiency of the integration (integrability) and the efficiency of the final product. Let us begin by evaluating the former.

7.3.1 Integrability

The evaluation of the integrability of the model is based on the complexity of the process of integration and on the range of systems into which the belief system can be integrated.

Complexity of the integration

The integration process is divided into a sequence of separate phases; the complexity of the entire process is the result of the composition of the complexities of the various phases. Let us examine each of them separately.

The very initial part of the process consists of converting the interpretation function $\mathcal{I}(\,)$ into the set of (acyclic) inference rules Ro, as explained in Theorem 5.1 (Composed Closure).

If the function $\mathcal{I}(\,)$ is defined simply as a list of explicit associations between propositions of $Lo \setminus Li$ and boolean functions of inner propositions, this procedure can be carried out automatically, and its complexity is simply *proportional* to the number of associations employed to define $\mathcal{I}(\,)$ entirely.

If the definition of $\mathcal{I}(\,)$ is not explicit, the conversion of the interpretation function $\mathcal{I}(\,)$ into the set of acyclic rules Ro will be assumed to be performed by the user.

The first algorithm applied to transform the BP-planning problem is A1. The part of A1 which appears to be the most computationally expensive is the calculation of $\Omega_1(I) = \mathbf{C}_{Ri} \cup \text{CONS\_}(\mathbf{C}_{Ri}(I) \setminus I)$. However, because of the property of acyclicity of the set Ri, the complexity of this calculation is simply *linear* in the number of propositions present in the set I.

In fact, the closure $\mathbf{C}_{Ri}(I)$ can be calculated directly as $I \cup K$, where

$$K = \{\, c \in Lo \mid c \notin I, \exists r \in Ri : r = (p \vdash c), p \in I \,\}$$

The set K contains only the collection of the consequences of the rules which have their premiss already in I, for the addition of K to the closure will not 'activate' any other rule.

Notice that the coefficient of proportionality which relates the number 'N' of operations necessary to calculate the closure of I to the number of propositions in I contains the cardinality of the set of rules Ri. In other words, there exists a constant 'k' such that

$$N \leq k \cdot (\#Ri) \cdot (\#I)$$

The set of inference rules R_i can be considered as a 'fixed' parameter of the problem. In fact, for the problem of discourse planning, the set of rules adopted should not be influenced by the dimension of the discourse, that is, by the dimension of the set I. Hence, the cardinality of Ri can be assumed to be constant, with respect to the number $\#I$.

Since the calculation of $\Omega_1(I)$ requires essentially only to identify the set K defined above, its computational load N_{Ω_1} can be estimated as

$$N_{\Omega_1} \leq k' \cdot (\#I)$$

where k' is a constant which depends on $\#Ri$.

The rest of the algorithm A1 applies the transformation Γ_1 to the set of operators Op. This procedure appends a list of conditional effects to each operator $O = (P, A, D)$; the length of this list is directly proportional to the number of propositions in the original add and delete lists A, D, since $\#\mathbf{C}_R(S) \leq k \cdot \#R \cdot \#S$ for any set S if R is a set of acyclic inference rules.

However, as for the set of inference rules, in this context the set of operators Op can be considered as a 'fixed' parameter of the problem. In fact, the complexity of the problem is proportional to the dimension of the belief set (and of the set of goals); even if the state grows, the set of operators available for planning a persuasive discourse remains, in general, unaltered. Since for *any* given initial set of operators Op there exists a number 'l' such that the length of any A or D list is never bigger than l, then the computational load required to transform a *single* operator will be always smaller than a certain constant, c.

Moreover, if the set Op is fixed, its cardinality $\#Op$ can also be considered

constant; consequently, the whole transformation Γ_1, requiring a number N_{Γ_1} of operations proportional to $c \cdot (\#Op)$, has a computational load limited by a constant k''.

In conclusion, a single application of the algorithm $A1$ to a BP-planning problem requires a number of operations N_{A1} which can be estimated as

$$N_{A1} = N_{\Omega_1} + N_{\Gamma_1} \leq k' \cdot (\#I) + k''$$

where $\#I$ is the cardinality of the set I, and the two constants k', k'' depend on the dimensions of the sets Ri and Op.

Notice that the algorithm A1 could be applied, in the initial phase, more than once, if the given set Ri is not already acyclic.

The analysis for the transformation $A2$, which is applied during the following phase of the process of the integration, is in every respect analogous to the above one[3]. However, it should be pointed out that the rules integrated by A2 are of a different kind; more precisely, they all have the form

$$r) \quad \neg (q_1 \vee \ldots \vee q_k) \vdash c$$

Hence, the truth of the boolean function representing the premiss requires, in the worst case, the check for the *absence*, in the set I, of k expressions. This means that a rule with k terms in the premiss might require $k \cdot (\#I)$ matches — instead of $\#I$, as in A1 — in order to be evaluated applicable.

Assuming the constant 'm' to be the maximum number of terms which appear in the premisses of the rules of the set Ro_a, the number of operations executed by A2 still grows linearly with the growth of the cardinality of the initial set I:

$$N_{A2} = N_{\mathbf{C}_{Ro_a}} + N_{\Gamma_2} \leq k_2' \cdot (\#I) + k_2''$$

where $k_2' \ \alpha \ m \cdot (\#Ro_a)$.

The new BP-planning problem in output from A2 will be re-transformed by A1, in the third and final phase, in order to integrate the rest of the rules of Ro in the form $p \vdash c$.

[3]Notice that also the set Ro, deduced from the interpretation $\mathcal{I}(\,)$, can be assumed to be an invariant parameter of the problem.

In conclusion, considering the 'preparatory' conversion of the interpretation function $\mathcal{I}(\,)$ performed either automatically (and therefore with a procedure of linear complexity) or 'by hand' (when the definition of $\mathcal{I}(\,)$ is not explicit), the whole process of integration of a BP algorithm into a planning system requires a number of operations N which does not grow more than *linearly* at the growth of the number of propositions present in the initial state I:

$$N \leq k \cdot (\#I)$$

Range of integrability

The second issue related to the evaluation of the integrability concerns the range of planning systems into which the belief system can be integrated.

Since the belief system is not actually 'integrated' into the planner, but simply 'pre-processed' into the planning problem through a transformation of the initial state and of the set of operators, there are basically no restrictions on the range of possible 'host' planning systems, so long as they offer the few standard characteristics which have been assumed, consisting, essentially, of operators with preconditions, add and delete lists. Hence, *any* planner which is able to solve a standard planning problem (as defined in Section 5.1.1) with 'P-A-D' operator schemes, be it partial-order, non-linear, or causal-link, can be adopted as the basis of the discourse planning system.

Notice that the presence of the P-A-D lists in the operators is required but does not constitute an 'upper' limit to the complexity of the formalism. In other words, the operators can be more sophisticated and still be suitable for the same process of 'integration'. For example, if the operator schemes contain, besides a precondition list, a list of *filters* (specifying conditions which must be verified for the application but which are not used for subgoaling), the process of transformation remains unaltered. This is because the precondition list of an operator is actually left unchanged by the process which the operators undergo.

Exploiting this property, it is also possible to use, for example, *abstract* operators (and, thus, adopt *hierarchical* planners) in which a scheme consists of a 'shell' — specifying a list of filters — and a 'body', containing a list of subgoals (cf., for example, Fox and Long's AbNLP [45]).

7.3.2 Efficiency of the final product

The evaluation of the efficiency of the entire system for discourse planning is based on the evaluation of the *impact* that the integration process has on the efficiency of the planner, and of the complexity of the process of *maintenance* (e.g., addition of new features, extension of the language, etc.) of the final product.

With regards to the impact of the integration, as the analysis of the computational complexity has revealed, the effects of the process consist essentially of *i*) extending the initial state I with new propositions (in both algorithms A1 and A2); *ii*) extending the add and delete lists of the operators (A2); and *iii*) adding lists of conditional effects — of length proportional to the dimension of the add and delete lists — to the operators (A1).

Although these effects undoubtedly increase the dimension of the search space for the specific problem considered, the *inherent* complexity of the planning problem is not altered. In fact, as Erol *et al.* point out in [35], if the problem specifies a *fixed* set of operators, "... for any given planning domain that can be described with STRIPS operators, the complexity of planning is at most in PSPACE, and [...] there exist such domains for which planning is PSPACE-complete". Moreover, Erol *et al.* also confirm that if the planning operators are extended to allow conditional effects, the overall complexity is not affected. This contradicts a widespread belief that planning with conditional operators is harder than planning with regular STRIPS operators. However, as they point out, "... conditional operators are useful only when we have incomplete information about the initial state of the world, or the effects of the operators [...]. Otherwise, we can replace the conditional operators with a number of ordinary STRIPS-style operators, to obtain and equivalent planning domain." [*ibid.*, p.84]

As far as the maintenance of the system is concerned, the main advantage offered by adopting a pre-processing approach lies in the higher level of modularity. The 'integrator' system for the transformation of the planning problem is completely separated from the planning system. This relieves the programmer from any need to understand and modify the code of the host planner, both initially and in case of subsequent releases of new versions. Moreover, the integrator module can be modified independently from the planning system adopted: in this way, the phase of testing and debugging is notably facilitated, allowing the output of the problem transformation to be examined directly and before any plan is searched for.

7.4 Efficiency of the Belief System

The efficiency of the belief system as a stand-alone module is evaluated in terms of run-time efficiency and maintenance efficiency. In order to evaluate the run-time efficiency, the computational complexity of the BP algorithm 'A', constituting the core of the belief system specified in Chapter 4, is analysed. An equivalent analysis can be carried out for the belief system actually adopted for the discourse planning system described in Chapter 6, which constitutes a simplified version of the algorithm A.

7.4.1 Run-time efficiency

The BP algorithm 'A' is composed of two main steps (see page 96):

1 Calculate $S' = \mathbf{C}_R(S)$;

2 result:= $\mathcal{I}(w) \mid_{S'}$.

The first step consists of the calculation of the deductive closure of the state S using the set R of inference rules. The second step uses the result of the first one to evaluate the truth of any given expression $w :: <WFE>$.

Let us begin by examining the complexity of step 1.

The specific inference rules adopted present a common characteristic: their *premisses* contain only one proposition. This feature allows the following simplified analysis: consider a state S containing m propositions; the closure S' can be calculated by matching each expression $p \in S$ against the premisses of each rule, and collecting the *consequences* of each rule which produced a successful match in an additional set C_1. Repeating this operation for each proposition of S will produce a final set C_1 containing all the possible consequences which can be derived from S.

The same process will have to be repeated for C_1, which might contain some proposition generating further consequences: C_1 will then produce a new set C_2, whose elements, in turn, will be matched against the rules to possibly produce a new set C_3, and so forth.

This process will terminate when a set C_n generates an empty set of consequences $C_{n+1} = \emptyset$. The termination is guaranteed by Theorem 4.1, since S can be assumed to be finite. The closure S' can eventually be obtained as the union

$$S' = S \cup C_1 \cup C_2 \cup \ldots \cup C_n$$

The total amount of operations N_1 required to calculate the set S' can be assumed proportional to the number of 'matches' performed to identify all the consequences. This number can be obtained as the product between the number of propositions contained in S' and the number r of inference rules present in R:

$$N_1 \ \alpha \ (\#S + \#C_1 + \#C_2 + \#C_3 + \ldots) \cdot r$$

where $\#C_i$ represents the *cardinality* of C_i. If we count the *maximum* number of different *direct* consequences (that is, obtained through a 'one-shot' derivation) which R can produce from any proposition, and call 'c' this number, since every expression $p \in S$ cannot generate more that 'c' consequences, we can put an upper limit to the number of consequences present in C_1:

$$\#C_1 \leq c \cdot m$$

where $m = \#S$. Also, $\#C_2 \leq c \cdot \#C_1 = c^2 \cdot m$. Hence, in general,

$$\#C_i \leq c^i \cdot m$$

Because of the specific characteristics of the rules adopted by the algorithm A, it can be shown that the *maximum* number of iterations of the process, for the given set R, is equal to three. In other words, $C_i = \emptyset$ for $i \geq 3$.

This can be seen by following, for each inference rule, the consequences which it can generate. For example, consider rule i_9 (and i_{17}, which has the same premiss), and suppose that the proposition p = HBEL(SBEL(SR(e))) belongs to S, with $e :: <Ei>$.

Because of i_9–i_{17}, the set C_1 will contain

$$C_1 = \{\text{HBEL(SBEL}(e)), \text{HBEL(SR}(e))\}$$

The first proposition of C_1 does not match any premiss (since $e :: <Ei>$); the second one matches i_5 and produces

$$C_2 = \{\text{HBEL}(e)\}$$

Finally, $C_3 = \emptyset$, as HBEL(e) does not 'activate' any other rule. The same analysis can be repeated for each inference rule in R.

Therefore, for the BP algorithm A, the number N_1 of operations required by step 1 can be evaluated as

$$N_1 \quad \alpha \quad (m + \#C_1 + \#C_2) \cdot r$$

that is

$$N_1 \leq k \cdot (1 + c + c^2) \cdot m \cdot r$$

where c represents the maximum number of different direct consequences which can be produced by any expression of $<WFF>$, and k a certain constant. This formula indicates that the complexity of step 1, for the specific BP algorithm adopted, is *linear* in both the number of propositions in the state S and the number of inference rules in R.

Let us now analyse the computational complexity of step 2 of the BP algorithm A, consisting of the evaluation of $\mathcal{I}(w) \mid_{S'}$.

The first part, concerning the simplification of the expression w, always requires a number of operations smaller that a certain constant. In fact, no $<WFE>$ expression will ever undergo more simplifications than a specific number[4], since the number of nested modalities which the syntax allows in an expression is limited, and the simplifications can either *eliminate* one modal operator from the expression or substitute one with a different one (notice that the simplifications do not produce infinite 'cycles' of substitutions).

A similar reasoning can be applied for the second part of step 2, containing the evaluation of the function *check*( ). The association between an expression w' and the final boolean function $check(w') = \mathcal{I}(w)$ is based exclusively on the analysis of the most 'external' levels of w', which are limited in number and combinations. All the possible cases are considered by the function *check*( ), and every expression is eventually evaluated as a boolean function of a set of inner language terms, as

[4]For the given simplifications, this number is three.

the formal definition of interpretation stated.

For example, from the definition of *check*( ), it follows

$$\begin{aligned} \mathit{check}(\mathrm{NOT}(\mathrm{SUND}(e))) &= B(e) \vee B(\mathrm{NOT}(e)) \\ \mathit{check}(\mathrm{HBEL}(\mathrm{NOT}(\mathrm{HBEL}(e)))) &= B(\mathrm{HUND}(e)) \vee B(\mathrm{HBEL}(\mathrm{NOT}(e)) \end{aligned}$$

Even though the number of terms defining the final boolean function varies according to the initial expression w', this number is always *smaller* than a certain constant k', because of the limited number of modal operators admitted by the syntax $<WFE>$.

There are, however, two exceptions to this rule. If the expression w' contains one of the predicate forms

$$\mathrm{HUNK}(s), \quad \mathrm{HBEL}(\mathrm{SUNK}(s))$$

with $s :: <Su>$, the associated boolean function will contain a number of terms equal to the number of events of type $<Ei>$ which are present in the argument s. This is due to the integration of Equation 4.3 (and following one) in the interpretation function. Inf fact, because of Equation 4.3, the boolean function associated with an expression such as

$$w = \mathrm{HUNK}(\ \mathrm{SUP}(\ \mathrm{NOT}(\mathrm{E}_1), \mathrm{SUP}(\mathrm{E}_2,\ \mathrm{SR}(\mathrm{E}_3))\ )\ )$$

is

$$\mathcal{I}(w) = \mathrm{HUNK}(\mathrm{E}_1) \vee \mathrm{HUNK}(\mathrm{E}_2) \vee \mathrm{HUNK}(\mathrm{E}_3)$$

Notice that the number of events $e :: <Ei>$ present in a support event increases *exponentially* with the number of *levels* of support nesting of the expression. However, only in the worst case it will be necessary to evaluate the truth of *all* of these terms: since the boolean function is a *disjunction* of propositions, its evaluation will terminate as soon as a term evaluated *True* is encountered.

The maximum number of terms present in the final boolean function associated to a query w can be used to estimate the maximum number N_2 of operations required for the evaluation of $\mathcal{I}(w)\ |_{S'}$. In fact, each term (at most) will have to be searched for in the current (completed) state S', and evaluated *True* or *False* according to the result of the search.

According to the previous analysis, if we exclude the two mentioned exceptions, the computational load for the evaluation of the function *check*( ) can be considered smaller than or equal to a number *proportional* to the number m' of propositions present in the state S':

$$N_2 \leq k_2 \cdot m'$$

where k_2 is a constant determined according to the maximum number of terms appearing in a boolean function associated to an expression w not containing the arguments a_1=HUNK(s), a_2=HBEL(SUNK(s)).

If the expression w does contain one of the two specific forms HUNK(s) or HBEL(SUNK(s)), with s::<Su>, the estimated number of operations required becomes

$$N_2 \leq k_2' \cdot 2^l \cdot m'$$

where k_2' is a constant and l is the level of support nesting of the argument 's' in w (e.g. s=SUP(E_1,E_2) has level $l = 1$).

Summarizing, for the steps 1 and 2 of the BP algorithm A the following computational complexities have been deduced:

1 to calculate S': $N_1 \leq k_1 \cdot m \cdot r$

2 to evaluate $\mathcal{I}(w) \mid_{S'}$: $N_2 \leq \begin{cases} k_2' \cdot 2^l \cdot m' & \text{if } w \text{ contains } a_1 \text{ or } a_2 \\ k_2 \cdot m' & \text{otherwise} \end{cases}$

where k' is a constant, m is the cardinality of the state S, m' is the cardinality of S' and r is the number of inference rules in R. The expression $k_1 = k(1 + c + c^2)$ — or the value of the constant c — could vary if the set of inference rules R is modified. It should be noticed that in the belief system adopted for the implementation of the discourse planning system of Chapter 6, the two exceptions which can give rise to exponential complexity have not been endowed in the BP algorithm. Hence, in such system, the run-time computational complexity is essentially linear in the number of propositions in the state and in the number of inference rules of R.

If repeated queries $w', w'', w''', \ldots$ are given to the belief prover while the state S is left unaltered, the calculation of the closure set S' does not need to be repeated every time, and the real computational load of the algorithm is represented by step 2. However, considering the generality of the formal definition of an inference rule $\mathcal{P} \vdash c$ (given in Chapter 3), in which $\mathcal{P}$ can be *any* boolean function of the

language, the process of calculation of the deductive closure S' could result to be much more complex, in presence, for example, of rules containing premisses with more than one term. Such type of inference rules would not be, in general, suitable for the application of algorithms A1 and A2, which 'pre-encode' the belief system into the operators and allow a *localized* updating of the state, as realized in the discourse planning system.

In such cases, it will be necessary either to develop more sophisticated algorithms of transformation (as described by Garagnani in [53]), or to develop a mechanism for the updating of the current closure $S' = \mathbf{C}_R(S)$ based on a system of *links* which keep track of the relations of *dependency* between different propositions of the closure S', as implemented — for testing purposes — in the actual realization of the belief system of Chapter 4.

7.4.2 Maintenance efficiency

The efficiency of the process of belief revision and reason maintenance depends on the type of belief system which the user chooses to implement. For example, if a foundational approach is adopted, the process of revision after the change of one attitude or the addition or deletion of a belief could involve, potentially, the reconsideration of the entire set of beliefs currently present in the state. The complexity of such revision would be proportional to the level of 'depth' of the change effected; the closer to the 'basis' of the system is the change, the bigger the propagation it will produce on the rest of the system.

On the other hand, if a coherence approach is adopted, the complexity of belief revision depends on the level of belief justification maintained in the system, which can be subjective and context-dependent. A low level of belief justification (that is, most of the beliefs of the system lack a supporting evidence) would reduce the procedure of revision to the minimal changes necessary to maintain the global coherence.

The use of the belief system in conjunction with a planner allows the definition of operators which can carry out the operations of belief revision on the current state. According to the specific contents of such operators, the process of revision will be either propagated through the system to all of the beliefs related to the changes or simply limited to the beliefs which are involved in the construction of the plan. The latter approach has been adopted for the actual implementation of the discourse planning system, in which the revisions are restricted to the beliefs of the system which are *relevant* to the discourse which is being planned. This method is in line

with the idea of 'local' coherence described in the work of Doyle [31], presented in Section 2.2.2.

Moreover, this approach is also in accordance with the main framework into which the discourse planning system has been assumed to fit, as described in Section 4.1. In such view, the process of discourse planning is seen as performed in a 'hypothetical' space, in which a temporary, limited representation of the beliefs is built, 'on the fly', from an underlying, larger knowledge base. After the effects that the discourse *actually* produces on the hearer's beliefs have been determined with certainty, the knowledge base will have to be updated accordingly. However, such process is not part of the activity of discourse planning in itself, as it has been intended throughout all this work.

7.5 Correctness

As required by the criteria specified in Section 2.5, the belief system implemented satisfies the formal specifications which have been given in Chapter 4.

It is interesting to notice that although such formal specifications require the beliefs of the speaker (and of the hearer model) not to be *openly* contradictory (cf. restrictions imposed on S_1–S_3, p. 87), that is, not to actually contain two beliefs ϕ and $\neg\phi$, the system developed allows the kind of *logical inconsistency* described by Konolige, also related to the concept of *locality* introduced by Doyle (see Section 2.2.2). In fact, this situation may occur when the set of beliefs contains two formulae ϕ and ψ such that a contradiction could be *derived*, but the inference rules are not strong enough to discover it; in other words, the agent is not *aware* of holding beliefs which result to be contradictory. Similarly, in the context of the integration of the belief system with a planner, the reasoning process which is carried out during planning performs only the deductions which are necessary for the construction of the discourse plan. Hence, the derivation of contradiction (or, in general, of new beliefs) is entirely *goal driven*, as one would expect when dealing with bounded-resources reasoning agents.

Section 4.3 has illustrated how the implementation has been actually realized, and how the correct functioning of the system has been verified through its performance on various crucial tests.

The correctness of the discourse planning system is based on the correctness of the belief system and on the formal proof of the correctness of the process of integration. The formal proof of the correctness of the process has been given

in Chapter 5. Although these elements are sufficient to guarantee the correct functioning of the system, empirical evidence of its validity has been given at the end of Chapter 6, where a number of discourse plans generated in response to different examples of discourse planning problems have been illustrated.

7.6 Conclusions and Further work

As stated in Section 1.3, the central problem addressed in this thesis consisted of the formalization, implementation and integration (with a planning system) of a belief system for persuasive discourse structure generation. The three main results which have been presented as a solution for this problem consist of:

1. a general *paradigm* for the specification of belief systems (not necessarily discourse-generation oriented), given in Chapter 3;
2. the formal specification and realization of a belief system (defined according to the given paradigm) for persuasive discourse planning, given in Chapter 4;
3. the description, formal proof of correctness and implementation of an algorithm (consisting of the pre-processing of the given planning problem) for the integration of any belief system built according to the paradigm requirements (Chapter 5).

Notice that a simpler version of the belief system described in Chapter 4 has been used to produce an *example* of a system for the automatic generation of persuasive discourse plans (Chapter 6).

The three main results have been evaluated in the preceding sections of this chapter, where it has been shown that the specific requirements of expressiveness and correctness set up in Section 2.5 have been met, and an indication of the efficiency of the solution proposed has been given. More precisely, because of the acyclicity of the inference rules used to calculate the closure of the initial set, the belief system integrated in the actual implementation presents *linear* run-time computational complexity. The process of integration itself presents, also, a complexity which is *linear* in the dimension of the initial state, if the set of operators and the characteristics of the belief system are considered as fixed parameters of the problem. Even when this is not so, and, for example, the number of operators, or the number of inference rules, or of premisses of an inference rule, is increased, the resulting computational load still undergoes simply a linear or polynomial growth of the constant of proportionality.

In the belief system defined in Chapter 4, because of the specific definition of function of interpretation $\mathcal{I}(\ )$, the complexity of the evaluation of two specific types of queries is *exponential* in the level l of support nesting of the argument of the query (see page 216); however, it seems plausible to hypothesize the existence of a certain maximum level of support nesting in the queries, so that the number of operations can be considered smaller than a constant. These considerations, nevertheless, underline the significant impact that the specific characteristics of the interpretation function have on the complexity of the process of query evaluation, as it would be natural to expect.

Finally, in relation to the impact that the process of transformation proposed has on the planning phase, it has been shown that the pre-processing of the initial BP-planning problem does not inherently increase the computational complexity of the planning problem. Moreover, the formal correctness of the transformation algorithm is based upon a set of hypotheses (summarized in Section 5.3.5) which do not constitute *necessary* conditions for the pre-processing of a general BP-planning problem: in other words, such hypotheses have been 'contingently' adopted for the specific algorithms A1 and A2, and could be relaxed if a more sophisticated transformation procedure were adopted (as indicated in Section 5.3.1).

One of the limitations of the integrator system implemented in Chapter 6 lies in the fact that it instantiates all the variables present in the set of operator schemes given in input, producing a ground set of operators on which the transformation algorithms A1 and A2 are then applied. However, these algorithms are suitable to be implemented also as a transformation of operators with variables: as suggested in Section 6.3, such a system could be actually realised if the formalism for the definition of the preconditions of operators allows the presence of codesignation and non-codesignation constraints.

With regard to possible developments of the present work, one of the features of the paradigm for the belief system specification consists of the fact that it has been defined 'on top' of an hypothetical 'belief base' representation having the concept of *event* as fundamental semantic unit (see Section 3.1). This characteristic allows the system implemented to be used in conjunction with a large variety of knowledge/belief base systems which adopt this kind of representation, such as event-based semantic networks and conceptual graph theories (e.g. [4] [149] [134] [133]).

Secondly, an important feature of the transformation algorithm described in Chapter 5 (already pointed out in the last part of Section 7.3.1) consists of the

fact that the precondition lists of the operators are left unchanged by the pre-processing phase. This leaves open the possibility of implementing more sophisticated discourse planning systems, employing, for example, *hierarchical* planners which allow the adoption of abstract operators, such as those developed by Reed in [118].

Thirdly, a natural extension of the discourse planning system described in Chapter 6 consists of actually realizing in NL the persuasive discourse specified by the plan produced. This will mean, in a first approximation, substituting each primitive event E_i with its corresponding semantic content (i.e., the proposition it represents), and using the 'meaning' of the operators to identify the 'cue words' which should be added to relate the different propositions of the discourse.

For example, the 'Attack(E_i, E_j)' step of any of the plans presented at the end of the previous chapter could be translated into something on the line of

> "Although you believe that E_j, the fact that E_i supports the thesis that $\neg$ E_j. Hence, this raises a question over whether or not it is the case that E_j."

Similarly, a 'Persuade(E_i, E_j)' could become

> "Since you believe that E_i, and also that this fact is a sufficient reason to believe that E_j, then you should also believe that it is true that E_j."

The primitive events E_i should obviously be replaced by the corresponding NL expressions. An analogous realization in NL is shown in [51].

However, if the discourse plan contains many steps, this simple substitution might be not completely satisfactory, as the global intentional structure of the discourse — composed of segments and sub-segments having specific communicative goals — would not have been conveyed to the audience. Even if such intentional structure does not need to be always included in the message, it constitutes a fundamental element for the process of replanning of part of the discourse, and a useful basis for dealing with the issues concerning the attentional state of the audience and the specific linguistic register adopted.

In order to identify the intentional structure underlying the specific plan produced it will be necessary to trace back the *links* between the various parts of the discourse plan and the way in which the communicative goals have been decomposed and achieved. Such links are not explicitly handled by the IPP planning

system, whereas they would be directly accessible in a *causal link* planner, such as DPOCL [153].

Taking this kind of 'surface' aspect into account during the actual NL realization of the discourse might involve the re-ordering of some of the 'parallel' sub-steps of the plan produced by IPP, or the insertion of specific cue words aimed at informing the audience of the decomposition of the discourse into various segments (generated, for example, by different communicative goals).

Focus and order have been considered in the design of the architecture presented by Reed in [118] (already discussed in Section 2.4.2), where the problem of the discourse structure generation and *refinement* has been tackled through the development of a *hierarchy* of discourse operators having decreasing level of abstraction. As mentioned above, such hierarchy could, in principle, be adopted for this framework, and transformed using the integrator system implemented. This would allow the operators, initial state and goals to be endowed with the belief system and language specified in Chapter 6, or with any other belief system built in accordance with the theoretical paradigm developed.

To conclude, this thesis has treated the problem of the automatic construction of the logical structure of persuasive discourses through the analysis, formalisation, implementation and integration of a belief system. The work presented does not pretend to be exhaustive, nor free from imperfections; however, it is hoped to represent a contribution towards the solution of some of the issues related to this problem. In particular, we feel that one of the main objectives of this research has been achieved: namely, the realisation of a framework for the implementation of discourse planning systems easy and natural to use, and able to efficiently integrate formal belief systems which are flexible and expressive enough to model characteristics that are typically human. Such characteristics include the ability to deal with uncertainty and inconsistency, to represent belief deduction, justification and grounding, and suitability for the integration in systems which make use of only limited amounts of resources.

Appendix A

Discourse Planning Operators

Persuade(?a,?b)
Prec: HBEL(SUP(a,b)), HBEL(a), HUND(b),
SBEL(SUP(a,b)), SBEL(a), SBEL(b)
Add: HBEL(b)
Del: HUND(b)

Spoil-sup(?a,?b)
Prec: HBEL(a), HBEL(b), NOT(HBEL(SUP(b,a))),
NOT(SBEL(a)), NOT(SBEL(SUP(b,a))),
NOT(HBEL(HR(a))), NOT(HBEL(SR(a)))
Add: HUND(a)
Del: HBEL(a)

Spoil-arg(?a,?b)
Prec: HBEL(a), HBEL(SUP(b,a)), NOT(HBEL(b)),
NOT(SBEL(a)), NOT(SBEL(b)),
NOT(HBEL(HR(a))), NOT(HBEL(SR(a)))
Add: HUND(a)
Del: HBEL(a)

Attack(?a,?b)
Prec: HBEL(b), HBEL(a), HBEL(SUP(a,NOT(b))),
NOT(SBEL(b)), SBEL(a), SBEL(SUP(a,NOT(b))),
NOT(HBEL(HR(b))), NOT(HBEL(SR(b)))
Add: HUND(b)
Del: HBEL(NOT(b))

Assert(?a)
Prec: HUNK(a), SBEL(a)
Add: HUND(a)
Del:

Assert-real(?a)
Prec: SR(a), HUNK(SR(a))
Add: HBEL(SR(a))
Del:

Bibliography

[1] Ackermann, R.J. (1972) *Belief and Knowledge*, Anchor, New York.

[2] Aiello, M., Albano, A., Montanari, U. (1976) *Teoria della Computabilita', Logica, Teoria dei Linguaggi Formali*, ETS, Pisa (Italy).

[3] Alchourrón, C.E., Gärdenfors, P., Makinson, D. (1985) "On the logic of theory change: Partial meet functions for contraction and revision", *Journal of Symbolic Logic*, 50:510–530.

[4] Ali, S.S., Shapiro, S.C. (1993) "Natural Language Processing using a propositional semantics network with structured variables", *Mind and Machines*, 3(4).

[5] Allen, J.F., Perrault, C.R. (1980) "Analyzing Intention in Utterances", *Artificial Intelligence*, 15:143–178.

[6] Allen, J.F., Shubert L.K., Ferguson, G., Heeman, P., Hwang, C.H., Kato, T., Light, M., Martin, M., Miller, B., Poesio, M., Traum, D.R. (1995) "The TRAINS project: a case study in building a planning agent", *Journal of Theoretical Artificial Intelligence*, 7:7–48.

[7] Anderson, C., Weld, D. (1998) "Conditional effects in Graphplan", *AIPS-98*, pp.44–53.

[8] Armstrong, D.M. (1973) *Belief, Truth and Knowledge*, Cambridge University Press, Cambridge.

[9] Asher, N. (1986) "Belief in Discourse Representation Theory", *Journal of Philosophical Logic*, 15:127–189.

[10] Asher, N., Lascarides, A. (1994) "Intentions and Information in Discourse", *Proceedings of the Meeting of the Association for Computational Linguistics (ACL-94)*, pp.34–41.

[11] Asher, N., Koons, R. (1993) "The revision of Beliefs and Intentions in a Changing World", in *Proceedings of the AAI Spring Symposium Series: Reasoning about Mental States: Formal Theories and Applications.*

[12] Asher, N., Munindar, S. (1993) "A Logic of Intentions and Beliefs", *Journal of Philosophical Logic*, 22(5):513–544.

[13] Austin, J.L. (1962) *How to Do Things with Words*, Oxford University Press, New York.

[14] Blair, J.A. (1996) "Notes for the Tutorial on Evaluating Arguments at the First International Conference on Formal and Applied Practical Reasoning (FAPR-96)", (unpublished manuscript), Bonn.

[15] Blum, A.L., Furst, M.L. (1997) "Fast Planning Through Planning Graph Analysis", *Artificial Intelligence*, 90:281–300.

[16] Bird, R., Wadler, P. (1989) *Introduction to functional programming*, Prentice Hall International.

[17] Bratman, M.E., Israel, D.J., Pollack, M.E. (1988) "Plans and resource-bounded practical reasoning", *Computational Intelligence*, 4:349–355.

[18] Brazier, F., Dunin-Keplicz, B., Jennings, N.R., Treur, J. (1997) "DESIRE: modelling multi-agent systems in a compositional formal framework", in Huhns, M., Singh, M. (eds), *International Journal of Cooperative Information Systems - Special issue on Formal Methods in Cooperative Information Systems*, vol.1.

[19] Brazier, F., Dunin-Keplicz, B., Treur, J., Verbrugge, R. (1996) "Beliefs, Intentions and DESIRE", *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Australia.

[20] Chapman, D. (1987) "Planning for Conjunctive Goals", *Artificial Intelligence*, 32(3):333–377.

[21] Chellas, B. (1980) *Modal Logic: An Introduction*, Cambridge University Press, Cambridge, England.

[22] Clark, D.A. (1990) "Numerical and symbolic approaches to uncertainty management in AI", *Artificial Intelligence Review*, 4:109–146.

[23] Cohen, P.R. (1983) "A computational model for the analysis of arguments", Technical Report CSRG-151, University of Toronto, Ontario.

[24] Cohen, P.R., Levesque, H.J. "Rational Interaction as the Basis for Communication", in Cohen, P.R., Morgan, J., Pollack, M.E., (eds), *Intentions in Communication*, MIT Press, Boston, pp.221–255.

[25] Cohen, P.R., Perrault, C.R. (1979) "Elements of a Plan-Based Theory of Speech Acts", *Cognitive Science*, 3:177–212.

[26] Cohen, P.R., Greenberg, M.L., Hart, D.M., Howe, A.E. (1989) "Trial by fire: Understanding the design requirements for agents in complex environmants", *AI Magazine*, 10(2):32–48.

[27] Devlin, K. (1991) *Logic and Information*, Cambridge University Press, Cambridge, England.

[28] Doyle, J. (1979) "A truth maintenance system", *Artificial Intelligence*, 12(2):231–272.

[29] Doyle, J. (1992) "Reason Maintenance and Belief revision: Foundations vs. Coherence Theories", in Gärdenfors, P., (ed), *Belief Revision*, Cambridge University Press, Cambridge, pp. 29–51.

[30] Doyle, J. (1994) "Reasoned assumptions and rational psychology", *Fundamenta Informaticae*, 20(1–3):35–73.

[31] Doyle, J. (1996) "Toward Rational Planning and Replanning", in Austin, T., (ed), *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, AAAI Press, Menlo Park, CA, pp. 130–135.

[32] Dung, P.M. (1995) "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games", *Artificial Intelligence*, 77:321–357.

[33] Ellis, B. (1979) *Rational Belief Systems*, Blackwell, Oxford.

[34] Elvang-Gøransson, M., Krause, P., Fox, J. (1993) "Dialectic reasoning with inconsistent information", *Proceedings of the Conference on Uncertainty in Artificial Intelligence UAI-93*, pp. 113–121.

[35] Erol, K., Nau, D.S., Subrahmanian, V.S. (1995) "Complexity, decidability and undecidability results for domain-independent planning", *Artificial Intelligence* 76:75–88.

[36] Etzioni, O., Lesh, N., Segal, R. (1994) "Building softbots for UNIX", in Etzioni, O., (ed), *Software Agens — Papers from the 1994 Spring Symposium (Technical Report SS-94-03)*, AAAI-Press, pp.9–16.

[37] Fagin, R., Halpern, J.Y. (1988) "Belief, Awareness, and Limited Reasoning", *Artificial Intelligence*, 34:39–76.

[38] Fikes, R.E., Nilsson, N.J. (1971) "STRIPS: A New Approach to the Application of Theorem Proving to Probelm Solving", *Artificial Intelligence*, 2:189–208.

[39] Fink, E., Yang, Q. (1997) "Automaticallyt selecting and using primacy effects in planning: theory and experiments", *Artificial Intelligence*, 89:285–315.

[40] Fisher, A. (1988) *The Logic of Real Arguments*, Cambridge University Press, Cambridge, England.

[41] Fogelin, R.J., Sinnott-Armstrong, W. (1991) *Understanding Arguments* (Fourth Edition), Harcourt Brace Jovanovich College Publishers.

[42] Forrest, P. (1986) *The Dynamics of Belief*, Blackwell, Oxford.

[43] Fox, J., Das, S. (1996) "A unified framework for hypothetical and practical reasoning (2): lessons from medical applications", in Gabbay, D., Ohlbach, H.J., (eds), *Practical Reasoning: Proceedings of the International Conference on Formal and Applied Practical Reasoning, FAPR-96 (Lecture Notes in AI Vol. 1085)*, Springer Verlag, Berlin, pp. 73–92.

[44] Fox, M. (1997) "Natural Hierarchical Planning using Operator Decomposition", in Steel, S., Alami, R., (eds), *Proceedings of the 4th European Conference on Planning (ECP-97)*, Springer Verlag, pp.195–208.

[45] Fox, M., Long, D. (1995) "Hierarchical planning using abstraction", *IEE Process-Control Theory and Applications*, 142(3):197–210.

[46] Fox, M., Long, D. (1996) "An Efficient Algorithm for Managing Partial Orders in Planning", *SIGART Bulletin*, 7(4):3–9.

[47] Freeman, J.B. (1991) *Dialectics and the Macrostructure of Arguments*, Foris, Dordrecht.

[48] Freese, J.H. (trans) (1926) Aristotle, *The Art of Rhetoric*, Heinmann: Loeb Classical Library, London.

[49] Gabbay, D. (1992) "'LDS - Labelled Deductive Systems, 7th Expanded Draft", Imperial College Technical Report, London.

[50] Garagnani, M. (1997) "Belief Modelling for Discourse Plans", *Proceedings of the 16th Workshop of the UK Planning and Scheduling SIG*, Durham, England.

[51] Garagnani, M., Long, D.P., Fox, M. (1998) "Belief Systems for Conflict Resolution", *Proceedings of the ECAI-98 Workshop on Conflicts Among Agents*, Brighton, England.

[52] Garagnani, M. (1998) "Belief Systems and Plans for Communication", *Proceedings of the 15th International Congress on Cybernetics*, Namur, Belgium.

[53] Garagnani, M. (1998) "Converting Inference Rules into Conditional Effects", *Proceedings of the 17th Workshop of the UK Planning and Scheduling SIG*, Huddersfield, England.

[54] Gärdenfors, P. (1988) *Knowledge in Flux: Modelling the Dynamics of Epistemic States*, Bradford Books, MIT Press, Cambridge, MA.

[55] Gärdenfors, P. (1990) "The dynamics of belief systems: Foundations vs. Coherence theories", *Revue Internationale de Philosophie*, 172:24–46.

[56] Gazen, B.C., Knoblock, C.A. (1997) "Combining the Expressivity of UCPOP with the Efficiency of Graphplan", *Proceedings of the 4th European Conference on Planning (ECP-97)*, Touluse, France, pp.221-233.

[57] Geissler, C., Konolige, K. (1986) "A resolution method for quantified modal logics of knowledge and belief", in Halpern, J.Y., (ed), *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, Morgan Kaufmann, San Mateo, CA, pp. 309–324.

[58] Gillies, D. (1996) *Artificial Intelligence and Scientific Method*, Oxford University Press.

[59] Girle, R.A. (1992) "Contradictory Belief and Logic", *Advances in Artificial Intelligence Research*, 2:23–36.

[60] Grice, H.P. (1957) "Meaning", *Philosophical Review*, 66:377–388.

[61] Grice, H.P. (1975) "Logic and Conversation", in Cole P., Morgan J.L., eds. *Syntax and Semantics*, Academic Press, NY, 3:41–58.

[62] Grosz, B.J., Pollack, M., Sidner, C. (1989) "Discourse", in Posner, M., (ed), *Foundations of Cognitive Science*, MIT Press, pp. 437–468.

[63] Grosz, B.J., Sidner, C.L. (1986) "Attention, Intentions and the Structure of Discourse", *Journal of Computational Linguistics*, 12(3):175–204.

[64] Grosz, B.J., Sidner, C.L. (1990) "Plans for Discourse", in Cohen P., Morgan J., Pollack M.E., eds. *Intentions in Communication*, MIT Press, Cambridge, MA, pp. 417–444.

[65] Haas, A. (1986) "A syntactic theory of belief and knowledge", *Artificial Intelligence*, 28(3):245–292.

[66] Harman, G. (1986) *Change in View: Principles of Reasoning*, MIT Press, Cambridge, Massachusetts.

[67] Heim, I. (1982) *The Semantics of Definite and Indefinite Noun Phrases*, Ph.D. thesis, University of Massachusetts.

[68] Hillebrand, K. (trans) (1891) Artur Shopenhauer *Two Essays*, London.

[69] Hintikka, J. (1962) *Knowledge and Belief*, Cornell University Press, Ithaca, New York.

[70] Hobbs, J.R. (1985) "On the coherence and structure of discourse", Technical report CSLI-85-37, Center for the Study of Language and Information, Stanford University.

[71] Hovy, E.H. (1988) "Planning Coherent Multisentential Text", *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL-88)*, pp. 163–169.

[72] Hovy, E.H. (1990) "Pragmatics and Natural Language Generation", *Artificial Intelligence*, 43:153–197.

[73] Hovy, E.H. (1991) "Approaches to the Planning of Coherent Text", in Paris, C.L., Swartout, W.R., Mann, W.C., (eds), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, Kluwer, pp.83–102.

[74] Hovy, E.H. (1993) "Automated discourse generation using discourse structure relations", *Artificial Intelligence*, 63:341–385.

[75] Hudak, P., Wadler, P. *at al.* (1992) "Report on the programming language Haskell, a non-strict purely functional language" (Version 1.2), *ACM SIGPLAN Notices*, 27(5).

[76] Janlert, L.-E., (1985) "Modeling change — the frame problem", in Pylyshyn, Z., (ed), *The Frame Problem and Other Problems of Holism in Artificial Intelligence*, Ablex, Norwood, New Jersey.

[77] Jones, C.E. (1994) *Dialogue Structure Models: An Engineering Approach to Machine Analysis and Generation of Dialogue*, Ph.D. thesis, Department of Computer Science, University of Durham, England.

[78] Jones, M.P. (1991) "An introduction to Gofer" (Version 2.20), Technical report Yale University, May 1991.

[79] Kamp, H. (1981) "A theory of truth and semantic representation", in Groenendijk, J.A.G., Janssen, T.M.V. and Stokhof, M.B.J., (eds), *Formal Methods in the Study of Language*, Mathematical Centre Tracts 136, University of Amsterdam, Amsterdam, pp. 277–322.

[80] Koehler, J., Nebel, B., Hoffmann, J., Dimopoulos, Y. (1997) "Extending Planning Graphs to an ADL Subset", *Proceedings of the 4th European Conference on Planning (ECP-97).*

[81] Konolige, K. (1982) "A first order formalization of knowledge and action for a multiagent planning system", in Hayes, J. E., Michie, D., Pao, Y., (eds), *Machine Intelligence 10*, Ellis Horwood Limited, Chichester, Englan, pp. 41–72.

[82] Konolige, K (1986) *A Deduction Model of Belief*, Pitman Publishing, London.

[83] Konolige, K (1986) "What awareness isn't: A sentential view of implicit and explicit belief (position paper)", in J.Y. Halpern, (ed), *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, Morgan Kaufmann, San Mateo, CA, pp. 241–250.

[84] Krause, P., Ambler, S., Elvang-Gøransson, M. Fox, J. (1995) "A Logic of Argumentation for Reasoning under Uncertainty", *Computational Intelligence*, 11(1):113–131.

[85] Kripke, S. (1963) "Semantical analysis of moodal logic", *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96.

[86] Kuhn, T.S. (1962) *The Structure of Scientific Revolutions*, University of Chicago Press, Chicago.

[87] Kyburg, H. (1961) *Probability and the Logic of Rational Belief*, Wesleyan University Press, Middletown, Connecticut.

[88] Lehmann, F. (1992) "Semantic Networks", *Computers Math. Applications*, 23(2–5):1–50.

[89] Levesque, H.J. (1984) "A logic of implicit and explicit belief", *Proceedings AAAI-84*, Austin, Texas, pp. 198–202.

[90] Levy, D.M. (1979) "Communicative goals and strategies: between discourse and syntax", in Givon, T., (ed), *Discourse and Syntax*, Academic Press, New York, pp. 183–210.

[91] Mann, W.C., Thompson, S.A. (1988) "Rhetorical Structure Theory: toward a functional theory of text organization", *Text*, 8(3):243-281.

[92] Maybury, M.T. (1992) "Communicative acts for explanation generation", *International Journal of Man-Machine Studies*, 37(2):135–172.

[93] Maybury, M.T. (1993) "Communicative Acts for Generative Natural Language Arguments", *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, AAAI Press, pp.357–364.

[94] McAllester, D., Rosenblitt, D. (1991) *Systematic Nonlinear Planning*, MIT AI Memo No. 1339.

[95] McCarthy, J., Hayes, P.J. (1970) "Some philosophical problems from the standpoint of artificial intelligence", in Meltzer, B., Michie, D., (eds), *Machine Intelligence*, 4, Edinburgh University Press, Edinburgh, pp.463–502.

[96] McDermott, D., Hendler, J. (1995) "Planning: What it is, What it could be, An introduction to the Special Issue on Planning and Scheduling", *Artificial Intelligence*, 76:1–16.

[97] Moore, J.D., Paris, C.L. (1989) "Planning Text for Advisory Dialogues", *Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-90)*, Helsinky, Vol.2:pp.276–281.

[98] Moore, J.D., Paris, C.L. (1993) "Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information", *Computational Linguistics*, 19(4):651–695.

[99] Moore, J.D., Pollack, M.E. (1992) "A problem for RST: The Need for Multi-Level Discourse Analysis", *Computational Linguistics*, 18(4):537–544.

[100] Moore, J.D., Swartout, W.R. (1991) "A reactive Approach to Explanation: Taking the User's Feedback into Account", in Paris C.L, Swartouot W.R., Mann W.C., eds. *Natural Language in Artificial Intellignece and Computational Linguistics*, Kluwer, Boston, MA, pp. 3–48.

[101] Morgenstern, L. (1987) "Knowledge preconditions for actions and plans", *Proceedings of the Tenth International Joint Conference on AI (IJCAI-87)*, Milan, Italy, pp. 867–874.

[102] Moser, M., Moore, J.D. (1996) "Towards a Synthesis of Two Accounts of Discourse Structure", *Computational Linguistics*, 22(3):409–419.

[103] Newell, A., Simon, H.A. (1963) "GPS, a program that simulates human thought", in Feigenbaum, E.A., Feldman, J., (eds), *Computers and Thoughts*, Mc-Graw Hill, New York.

[104] O'Keefe, D.J. (1977) "Two concepts of argument", *The Journal of American Forensic Association*, 13:121–128.

[105] Parsons, S., Jennings, N.R. (1996) "Negotiation Through Argumentation — a Preliminary Report", *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-96)*, pp. 592–596.

[106] Pednault, E.P.D. (1989) "ADL: Exploring the middle ground between STRIPS and the situation calculus", *Proceedings of the Knowledge Representation Conference (KR-89).*

[107] Penberthy, J.S., Weld, D. (1992) "UCPOP: A Sound, Complete, Partial-order Planner for ADL", *Proceedings of the International Workshop on Knowledge Representation (KR-92)*, pp.103–114.

[108] Perelman, C., Ohlbrechts-Tyteca (1969) *The New Rhetoric*, University of Notre Dame Press, Paris.

[109] Perlis, D. (1985) "Languages with self reference I: Foundations", *Artificial Intelligence*, 25:301–322.

[110] Perlis, D. (1988) "Languages with self reference II: Knowledge, belief, and Modality", *Artificial Intelligence*, 34:179–212.

[111] Polanyi, L. (1988) "A formal model of the strucure of discourse", *Journal of Pragmatics*, 12:601–638.

[112] Pollack, M.E., Ringuette, M. (1990) "Introducing the Tileworld: Experimentally evaluating agent architectures", *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, pp.183–189.

[113] Pollock, J.L. (1987) "Defeasible Reasoning", *Cognitive Science*, 11:481–518.

[114] Pollock, J.L. (1994) "Justification and defeat", *Artificial Intelligence*, 67:377–407.

[115] Popper, K.R. (1959) *The Logic of Scientific Discovery*, Hutchinson, London.

[116] Prakken, H. (1996) "Dialectical proof theory for defeasible argumentation with defeasible priorities", *Proceedings of the FAPR-96 Workshop on Computational Dialectics*, Bonn.

[117] Rao, A.S., Georgeff, M.P. (1991) "Modeling rational agents within a BDI-architecture", in Fikes, R., Sandewall, E., (eds), *Proceedings of Knowledge Representation and Reasoning (KR& R-91)*, Morgan Kaufmann, San MAteo, CA, pp.473—484.

[118] Reed, C. (1998) *Generating Arguments in Natural Language*, Ph.D. thesis, University College London.

[119] Reed, C., Long, D. (1997) "Multiple subarguments in logic, argumentation, rhetoric and text generation", *Proceedings of the International Joint Conference on Quantitative and Qualitative Practical Reasoning*, Springer Verlag, Berlin.

[120] Reed, C., Long, D. (1997) "Persuasive Monologue", in *Proceedings of the OSSA Conference on Argument and Rhetoric*, St. Catherines, Canada.

[121] Reed, C., Long, D., Fox, M. (1996) "An Architecture for Argumentative Discourse Planning", in Gabbay D., Ohlbach H.J., eds. *Practical Reason: Proceedings of the International Conference on Formal and Applied Practical Reasoning, FAPR'96 (LNAI 1085)*, Springer Verlag, pp. 555–565.

[122] Reiter, R. (1978) "On closed world data bases", in Gallaire, H., Minker, J., (eds), *Logic and Data Bases*, Plenum Press, New York, pp.55-76.

[123] Reed, C., Fox, M., Long, D.P., Garagnani, M. (1996) "Persuasion as a Form of Inter-Agent Negotiation", *Lecture Notes in Artificial Intelligence*, 1286:120–136.

[124] Reichgelt, H. (1989) "Logics for reasoning about knowledge and belief", *Knowledge Engineering Review*, 4(2):119–139.

[125] Ryle, G (1950) " 'If', 'So' and 'Because' ", in Black, M., (ed), *Philosophical Analysis: A Collection of Essays*, Cornell University Press, Ithaca, NY, pp. 323–340.

[126] Sacerdoti, E. (1974) "Planning in a Hierarchy of Abstraction Spaces", *Artificial Intelligence*, 5:115–135.

[127] Sacerdoti, E. (1975) "The Nonlinear Nature of Plans", *Proceedings of the fourth International Joint Conference on AI (IJCAI-75)*, pp.206–214.

[128] Searle, J.R. (1969) *Speech Acts*, Cambridge University Press, Cambridge.

[129] Scruton, R. (1995) "A short history of modern philosophy", Routlege, London.

[130] Shapiro, S.C. (1971) "A net structure for semantic information storage, deduction and retrieval", in *Proceedings of the Second International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, California, pp. 512–523.

[131] Shapiro, S.C. (1993) "Belief paces as sets of propositions", *Journal of Experimental AI*, 5:225–235.

[132] Shiu, S., Luo, Z., Garigliano, R. (1996) "Type Theoretic Semantics for SemNet" in Gabbay, D., Ohlbach, H.J., (eds), *Practical Reasoning: Proceedings of the International Conference on Formal and Applied Practical Reasoning, FAPR-96*, (Lecture Notes in AI Vol. 1085), Springer Verlag, pp.582–595.

[133] Smith, M.H., Garigliano, R., Morgan, R.C., Shiu, S., Jarvis, S. (1994) "LOLITA: A Natural Language Engineered System", unpublished manuscript.

[134] Sowa, J.F. (1984) *Conceptual Structures: information processing in mind and machine*, Addison-Wesley.

[135] Stefik, M. (1981) "Planning with constraints (MOLGEN:Part 1)", *Artificial Intelligence*, 16(2):111–139.

[136] Stefik, M. (1981) "Planning with constraints (MOLGEN:Part 2)", *Artificial Intelligence*, 16(2):141–169.

[137] Sussman, G.J. (1975) *A computer Model of Skill Acquisition*, MIT Press, Cambridge, Massachusetts.

[138] Spencer-Smith, R. (1991) "Modal Logic", *Artificial Intelligence Review*, 5:5–34.

[139] Stalnaker, R. (1984) *Enquiry*, MIT Press, Cambridge, Massachusetts.

[140] Tate, A. (1977) "Generating Project Networks", *Proceedings of the International Joint Conference on AI (IJCAI-77).*

[141] Thomason, R. (1980) "A note on syntactical treatments of modality", *Synthese*, 44:391–395.

[142] Toulmin, S.E. (1958) *The Uses of Argument*, Cambridge University Press, Cambridge, England.

[143] Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., Blythe, J. (1995) "Integrating Planning and Learning: The PRODIGY architecture", *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1).

[144] Vreeswijk, G. (1992) "Reasoning with Defeasible Arguments", in Wagner, G., Pearce, D., (eds), *Proceedings of the Euroopean Workshop on Logics in AI (JELIA-92)*, Springer Verlag, Berlin, pp. 189–211.

[145] Walker, M.A., Rambow, O. (1994) "The role of Cognitive Modelling in Achieving Communicative Intentions", in *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport, Maine.

[146] Wilkins, D.E. (1984) "Domain-independent planning: Representation and plan generation", *Artificial Intelligence*, 22(3):269–301.

[147] Wilson, B.A. (1980) *The anatomy of Argument*, University Press of America, Washington, D.C.

[148] Woods, S. (1993) *Planning and Decisioon Making in Dynamic Domains*, Ellis Horwood, Chichester, England.

[149] Woods, W.A., Schmolze, J.G. (1992) "The KL-ONE family", *Computers Mathematics and Applications*, 23(2).

[150] Wooldridge, M. (1992) *The Logical Modelling of Computational Multi-Agent Systems*, PhD Thesis, Department of Computation, UMIST, Manchester, UK.

[151] Wooldridge, M, Jennings, N.R. (1995) "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review*, 10:115-152.

[152] Young, R.M. (1996) "A developer's Guide to the Longbow Discourse Planning System", University of Pittsburgh, ISP TR-96-2.

[153] Young, R.M., Moore, J.D. (1994) "DPOCL: A principled Approach to Discourse Planning", in *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport, Maine, June 1994.

[154] Young, R.M., Moore, J.D., Pollack, M.E. (1994) "Towards a Principled Representation of discourse Plans", *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA.

[155] Young, R.M., Moore, J.D., Pollack, M.E. (1994) "Decomposition and Causality in Partial Order Planning", *Proceedings of the Second International Conference on AI and Planning Systems (AIPS-94)*, Chicago, IL.